Environmental Multiway Data Mining

Jérémy E Cohen

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES

Spécialité : **Électronique, électrotechnique, automatique et traitement du signal**

Arrêté ministériel : 7 août 2006

Présentée par

Jérémy COHEN

Thèse dirigée par **Pierre COMON**

préparée au sein du laboratoire **GIPSA-lab, CNRS**
dans **l'École Doctorale EEATS**

Fouille de données tensorielles environnementales

Thèse soutenue publiquement le **5 Septembre 2016**,
devant le jury composé de :

M. David BRIE
Professeur au CRAN à Nancy, Président du jury
M. Laurent ALBERA
Maître de conférence à l'Université de Rennes à Rennes, Rapporteur
M. Lieven DE LATHAUWER
Professeur à KU LEUVEN à Leuven, Rapporteur
M. Nikolaos D. SIDIROPOULOS
Professeur à l'Université du Minnesota à Minneapolis, Rapporteur
M. Pierre COMON
Directeur de recherche au GIPSA-lab, CNRS à Grenoble, Directeur de thèse


THESIS

To obtain the title of

PhD OF THE UNIVERSITE OF GRENOBLE ALPES

Speciality : **Electrical engineering, Automatic control and Signal rocessing**

Defended by

Jérémy E. COHEN

Thesis advisor: **Pierre COMON**

prepared at **GIPSA-lab, CNRS**
within the Graduate School **EEATS**

Environmental Multiway Data Mining

Thesis publicly defended on the **5th September 2016**,
in front of the jury composed by :

Mr. David BRIE
Professor at CRAN in Nancy, President of the jury

Mr. Laurent ALBERA
Associate Professor at the Université de Rennes in Rennes, Rapporteur

Mr. Lieven DE LATHAUWER
Professor at KU LEUVEN in Leuven, Rapporteur

Mr. Nikolaos D. SIDIROPOULOS
Professor at the Université of Minnesota in Minneapolis, Rapporteur

Mr. Pierre COMON
Research Director at GIPSA-lab, CNRS in Grenoble, Thesis advisor

Abstract

English Abstract Among commonly used data mining techniques, few are those which are able to take advantage of the multiway structure of data in the form of a multiway array. In contrast, tensor decomposition techniques specifically look for intricate processes underlying the data, where each of these processes can be used to describe the multilinear structure of the data array. The work reported in the following pages aims at incorporating various external knowledge into the tensor canonical polyadic decomposition, which is usually understood as a blind model. The first two chapters of this manuscript introduce tensor decomposition techniques making use respectively of a mathematical and an application framework. In the third chapter, the many faces of constrained decompositions are explored, including a unifying framework for constrained decomposition, some decomposition algorithms, compression and dictionary-based tensor decomposition. The fourth chapter discusses the inclusion of subject variability modeling when multiple arrays of data are available stemming from one or multiple subjects sharing similarities. State of the art techniques are studied and expressed as particular cases of a more general flexible coupling model later introduced. The chapter ends on a discussion on dimensionality reduction when subject variability is involved, as well as some open problems.

French Abstract Parmi les techniques usuelles de fouille de données, peu sont celles capables de tirer avantage de la complémentarité des dimensions pour des données sous forme de tableaux à plusieurs dimensions. A l'inverse, les techniques de décomposition tensorielle recherchent spécifiquement les processus sous-jacents aux données, qui permettent d'expliquer les données dans toutes les dimensions. Les travaux rapportés dans ce manuscrit traitent de l'amélioration de l'interprétation des résultats de la décomposition tensorielle canonique polyadique par l'ajout de connaissances externes au modèle de décomposition, qui est par définition un modèle aveugle n'utilisant pas la connaissance du problème physique sous-jacent aux données. Les deux premiers chapitres de ce manuscrit présentent respectivement les aspects mathématiques et appliqués des méthodes de décomposition tensorielle. Dans le troisième chapitre, les multiples facettes des décompositions sous contraintes sont explorées à travers un formalisme unifié. Les thématiques abordées comprennent les algorithmes de décomposition, la compression de tenseurs et la décomposition tensorielle basée sur les dictionnaires. Le quatrième et dernier chapitre présente le problème de la modélisation d'une variabilité intra-sujet et inter-sujet au sein d'un modèle de décomposition contraint. L'état de l'art en la matière est tout d'abord présenté comme un cas particulier d'un modèle flexible de couplage de décomposition développé par la suite. Le chapitre se termine par une discussion sur la réduction de dimension et quelques problèmes ouverts dans le contexte de modélisation de variabilité sujet.

To my family and friends, for being an endless source of inspiration.

Acknowledgements

Contents

A short and non-technical introduction

For a few years now, people have been asking me the same question again and again: What have I done during my PhD ? My answer heavily depends on who is asking the question. There has to be four or five layers of answers, ranging from a very technical explanation of tensor algebra to a simplified picture of what is data processing and why it is important. But whatever the level of complexity of the explanations, I always felt the topics covered in my research were not easy to explain. For a specialist, the wide variety of notations and tricky concepts within multilinear algebra may hinder mutual comprehension while for an amateur, simply providing some context seems close to impossible. At least in the few minutes of attention people usually pay before getting bored. Even though this is no easy task, one of the goals in writing this manuscript is to explain to a wide audience how tensors can be used in a simple but rigorous manner, also by simplifying the tools and concepts where it is possible.

The first two chapters of this manuscript respectively deal with the mathematical framework needed to understand multilinear algebra, and with some real life applications which help picturing the challenges of multiway array processing. But if possible, I would like anybody to be able to understand at least the main ideas I have been working on during these three years, and since chapters 1 and 2 are written for fellow researchers, this introduction serves the purpose of presenting my work to people who are not familiar with mathematics, informatics or signal processing.

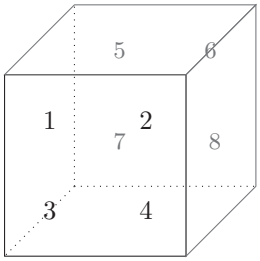
Let us start with discussing what data scientists (*a.k.a.* signal processing researchers) work with: "data". In a very practical way, data can be thought of as a collection of numbers. For instance, if one is interested in demographics, the population of Paris, Lyon and Marseille in 2010 is approximately given in millions by the following numbers: $[2.240\ 0.484\ 0.850]$. The process of collecting data is called measurement, observation, data collection or survey depending on the kind of data being acquired. Of course data have a wider meaning that just a collection of numbers, and any information obtained from any event can be considered as a datum. For example, our eyes keep on recording the scenes around us, thus collecting data on our surroundings which are not stored as a collection of number by our brains. But since data scientists use mathematics and informatics as tools, unlike human brains, working with numbers is mandatory for us.

Data collection is a very wide research topic, which is highly dependent on the kind of data to be acquired. Astrophysicists, chemists, doctors, researchers in telecommunications, sociologists and many more have data collection at the center of their occupations. It is capital to understand to some extend the physical properties of what is being measured in order to obtain meaningful data. Say one is again interested in French demographics, then estimating the population of Calcutta, London and Osaka may not be quite interesting. Also, measuring the temperatures in Paris, Lyon and Marseille does not bear meaningful information on the demographics of these towns.

Once data have been collected and stored in a data set, it may be handed over to data scientists who try to extract information from these data. A fact that is misunderstood by non-scientists while being at the core of data science is that at this stage, the data can be processed with methods designed for *any* data set, regardless of where it comes from. So what data scientists really do is twofold. First, they design mathematical tools with good properties, meaning that they build their tools to be efficient and as little costly as possible. Think of a metalsmith who needs good and various hammers to work with different metals. Second, they apply these tools on data that have been provided, and try to understand what is obtained as a result. So when a data scientist says that he extracts information from data sets, what is to be understood is that he applies some general tools to some collected data, but obtains something which is interpreted differently for each tool and each kind of data set.

This is why social scientists use the same kind of methods for studying the results of their surveys than chemists to identify the components in a chemical mixture. These methods are also used for instance in image processing, video surveillance and radar. Data science is a form of applied mathematics, but the applied part is generally not what really drives researchers in this area of research. What we like is to build tools made of mathematics. Sometimes we think about what kind of data it would be nice to use it on, and sometimes we don't. Some tools are specific to one application because there was a specific need for it, and applying it to this application yields easily interpreted results, just like the brain treats the data collected by our eyes to create what we see. On the other hand, some tools are made so general that they mine any kind of data with nice-looking results, which is only possible because the laws of nature can mostly be described by the same mathematical formalism, like Galileo suggested in his time.

The methods I work with, tensor decomposition methods, are of the latter kind, although they do not exactly apply to any kind of data. The data I work with have to be arrays of strictly more than two dimensions, meaning that the numbers can be rearranged in the following format:



Here the cube of numbers is of size 2 by 2 by 2, but what is required is that it is a bloc with more than two ways, so it can be of size anything by something-else by another-thing. This also means that any data set containing more than eight numbers is a candidate for tensor decomposition methods.

Now because a metalsmith can use his hammer on some metal does not mean this will be an efficient process, and what he needs to know to be a good metalsmith is what kind of metal he can *efficiently* use his hammer on. The same goes for data scientists using tensor decomposition methods. Tensor decomposition methods can be used on a wide panel of data set, but there are much fewer data sets onto which it can be applied while bearing satisfactory results. They need to satisfy an additional constraint.

The data should be well described by a small number of 'simpler' data sets. While the notion of 'simple' data set is understood formally as a mathematical property called the rank, let us illustrate this constraint on the practical example of demographic study of France. Suppose this time researchers have provided a data set containing the population of all towns in France measured each year from 1945 to 1980. It is well known that around that time, many people left the countryside to find work in the growing cities. This means two processes underlying the data may be identified: the drop in the population of small country towns over time, and the increase in the population of major towns over time. By extracting the population numbers related to the small towns, it is possible to build another data set containing only the population of small French towns between 1945 and 1980. But is it a 'simpler' data set ? It is simpler in the way that is understood in this manuscript only if the decrease of population in all of these small towns is comparable, that is the populations decrease approximately at the same rate at everyplace it decreases. Then the extracted data set of population in the small towns can be summarized by the knowledge of this decrease rate over time, and therefore is 'simple'. In short, a data set is simple if it can be summarized by a few well chosen numbers, here the amount of decrease of population over time shared by all small towns. This is illustrated by the figure below.

Note that in this example the data are not considered to be a cube, it hence cannot be studied using tensor decomposition methods. If it were, then what tensor decomposition does is to extract the 'simpler' data from the original cubic data set. In the above example, tensor decomposition would extract the two trends (increase and decrease) in population as well as which towns they refer to (small or large). But the knowledge of which town is small and which is large is not required, which makes tensor decomposition techniques very useful when little is known on the simpler processes that generated the data. This last point is important, because it means that when a data scientist has acquired some data which would be efficiently processed by tensor decomposition, he or she does not need to know anything about the physical meaning of the numbers inside his

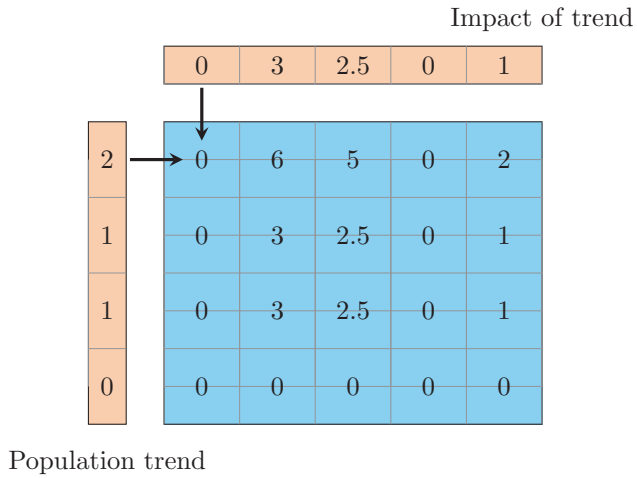


Figure 1: A simple data set (in blue) that can be represented by two sets of numbers (in salmon). In the demographic example, these numbers stand respectively for a trend in population (decrease) and a coefficient city per city that corresponds to the impact of the trend on city population. Since the trend here is a decrease in population, the zeros in the above set characterize major cities where demographics is not dropping.

cube, or where the data come from.

Now that the tool used in my work and the material onto which it can be used as well as what can be crafted is specified, it is time for what really motivates the writing of the hundred or so following pages, that is, my humble contributions. They have been mostly twofold. In both research topics quickly described below, the core idea I wanted to develop was that data are not simply a collection of random numbers, and that the simpler structure that is to be extracted has some physical interpretation which can be useful for the data processing. For instance when obtaining data in chemistry (and more specifically in fluorescence spectroscopy), the acquired numbers are always positive. Moreover, the simpler data in this context can be shown to be related to the concentrations of different chemical components. Concentrations are always positive because they stand for an amount of something. When using a signal processing tool to extract the simpler structure, knowing the result should be positive helps with extracting the concentrations of the chemicals, just like our metalsmith performs much better at crafting when he can mentally anticipate the results of his hammering with respect to when he cannot. In short, what I aimed for in my PhD thesis was to use the very general tool known as tensor decomposition while taking into account application-related external information that improves the performance of the method.

First, I worked on what is called dimensionality reduction. Very practically, the goal of dimensionality reduction is to reduce the sizes of the data while leaving the important information contained inside unaltered. It is feasible if a lot of data are collected while the number of simple data sets is small. For instance, if the decrease of population in the previous example happens at a steady rate, then on a plot where population is reported over time, what is obtained is a straight line with negative slope. Since only two points are necessary to plot a straight line, then in theory only two numerical values are necessary to summarize the temporal evolution of the population in small towns. Two values of population each at one point in time summarize all the values that were originally measured every year for sixty years. Of course many people have already designed signal processing tools to compute this dimensionality reduction, which can be understood as a preprocessing before the actual tensor decomposition. But what had not been done before was to take into account the additional information available on the data after the dimensionality reduction had been computed.

Second, I worked on the simultaneous extraction of useful information from multiple data sets which are somehow related. For instance, if the demographic evolution of Germany between 1945 and 1980 is provided alongside with the data of our previous example, then because rural exodus also happened in Germany, similar information is hidden in both data sets and can be mined simultaneously to increase the accuracy of the methods. The main difficulty here is to describe the connection the multiple data sets share, since some data sets may not be related, or related in a

way that is difficult to incorporate in tensor decomposition techniques.

If by reading this introduction a novice (or not so novice) reader could get a glimpse of my works, and if by any chance he wishes to know more on the topic of tensors, then he or she can either read the hundred pages or so that follow this non-technical discussion, or find some complementary information in one of the following books [16, 57, 101]. I also apologise for all the topics that could not be covered in this manuscript, since tensor decompositions cover such a wide range of topics. Those include telecommunications, sensor arrays, differential equations, audio processing, algebraic geometry, social sciences, psychometry, rating predictions.

Contributions

Work conducted during the three years of PhD led to the following contributions.

Major contributions

- A nonlinear decomposition model was designed and studied for fluorescence spectroscopy data when the concentration of the components is high. This collaboration with Xavier Luciani lead to a journal publication in Chemometrics and Intelligent Laboratory Systems [34] and a conference paper at EUSIPCO (2014) [29].
- An algorithm that allows to decompose compressed tensors that are constrained in the uncompressed domain called PROCO-ALS was proposed in the IEEE Signal Processing Letters [32] and presented at ICASSP (2015). Extended results were also presented at GDR ISIS meeting of 06/08/2016.
- To improve the modeling of joint decomposition of tensors, a flexible coupling for data fusion involving tensors was developed in collaboration with Rodrigo Cabral Farias. Results were first communicated at the LVA/ICA (2015) conference [22] and then led to a journal publication in the IEEE Transactions on Signal Processing [21]. These results were presented informally at two workshops, JODA (2015) and TDA (2016). Additionally, an application of flexible couplings to the fusion of EEG and Gaze data led to a conference paper at SAM (2016) [99].
- A new application of tensor decomposition methods, spectral unmixing, was explored in collaboration with Miguel Veganzones. Results were promising and reported in several publications: one journal publication in IEEE Transactions on Geoscience and Remote Sensing [116] for the nonnegative decomposition of hypersepctral movies, and three conference papers at EUPSICO (2015) [117], EUSIPCO (2016) [118] and WHISPERS (2016) [119] respectively tackling the angle, patch and angle diversity.

Minor contributions

- Joint dimensionality reduction for coupled data sets under various scenarios. This led to one conference paper at EUSIPCO (2016) [33] and one national conference paper at GRETSI (2015) [31].
- How to efficiently use dictionaries in a tensor decomposition model is ongoing work, but some preliminary results are contained in a submitted journal paper [35].
- Although notations habits are quite personal and subjective, I tried to create a formalism for multiway array processing that could be related to how tensors are defined in mathematics, and how they are used in quantum physics. This is discussed in a short paper on arxiv [30] and a submitted paper on a formalism for constrained decomposition [35].
- A fairly important topic to which I contributed very slightly is the derivation of Cramér-Rao Bounds for structured tensors, reported in a conference paper at ICASSP (2015) [9].

Personally I do not think that these numbers matter, but since nowadays they are used as a performance index, note that contributions add up to 4 international journal papers plus 1 submitted, 8 international conference papers, 1 national conference papers, 2 contributions for workshops without proceedings and 1 arxiv paper. As first author, these numbers reduce to 2 international journal papers plus 1 submitted, 2 international conference papers and 1 national conference paper, 2 contributions for workshops without proceedings and 1 arxiv paper.

Part I

An introduction to tensor decompositions and their uses

Chapter 1

Multilinear tools and computation

Summary This first chapter deals with formalism, notations and basic tools in multilinear algebra. Definitions and properties are given for real three way arrays and more generally for third order tensors, but a generalization of notations to higher orders is trivial. It is structured as follows:

- Tensors understood as multiway arrays, allowing simple but narrow definitions.
- Tensors as described in multilinear algebra.
- Computation rules explaining how to manipulate tensors with ease.
- Introducing the models that will be used all along this manuscript, *i.e.* Canonical Polyadic Decomposition and Tucker decomposition.

This chapter is meant as both a technical introduction to the manuscript as well as a contribution to clarify the manipulation of tensors, which becomes natural given some proper notations.

Introduction Not everyone in the tensor community defines and manipulates tensors in the same fashion. Data scientists tend to refer to multiway arrays and make use of notations from Harshman or Kiers [59, 70], or more recently, from Kolda [72] or Comon [37]. On the other hand, mathematicians interested in the numerical and theoretical treatment of tensors define tensors as vectors from tensor spaces, which will be defined below. This formalism is closer to the historical definitions given by Whitney and Bourbaki [11, 122].

The reasons for this division may be twofold. For data scientists, tensor decompositions stem from regression models [64], where the tensor product concept was not initially needed. Moreover, the use of more intricate tensor notations is difficult to grasp in a general context. However as we show in this chapter, notations can be made quite natural for finite dimension tensors. Multiway arrays are simply tensors expressed in a particular instance of isomorphic tensor spaces.

For further readings about possible formalisms, Bro's thesis [16] gives most tools necessary for multiway array processing, while Hackbush's book [57] provides comprehensive definitions and properties for tensor product spaces. A small paper grouping some tensor formalism concepts with useful computation rules is available on arxiv [30].

1.1 Multiway formalism

Multiway array processing refers to mining data from tables of more than two dimensions. As a first naive approach, manipulation of such tables does not require the multilinear algebra tools that are introduced in the next section. In all the discussion below, we consider real arrays.

1.1.1 Some basic definitions

First let us provide a definition of a three way array.

Definition 1 *A three way array $\mathcal{T}$ of size $K \times L \times M$ is an element of the finite-dimension real vector space $\mathbb{R}^{K \times L \times M}$. It is uniquely defined by the set of its coordinates T_{klm} where $1 \leqslant k \leqslant K$, $1 \leqslant l \leqslant L$ and $1 \leqslant m \leqslant M$.*

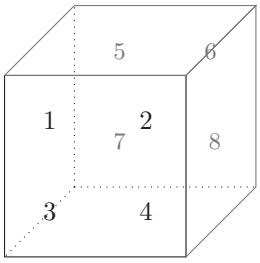


Figure 1.1: A three way array of dimension $2 \times 2 \times 2$

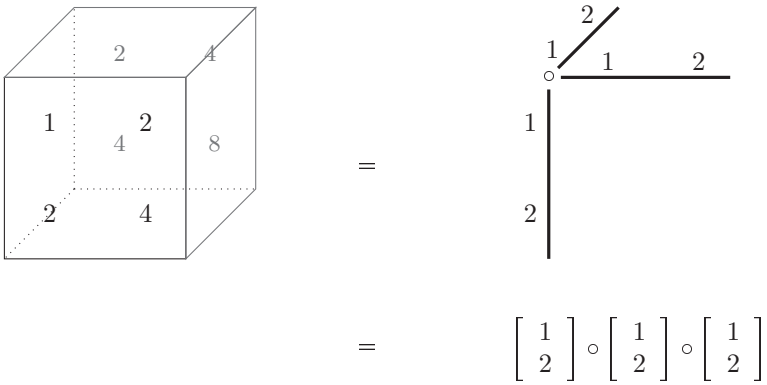


Figure 1.2: The outer product of three vectors of size 2

In other words, a three way array is a table $\boldsymbol{\mathcal{T}}$ with three indices k, l and m, so that it contains KLM entries (see Figure 1.1). Different words can be used for addressing the ways of an array: *ways*, *modes*, *diversities*. The word diversities in particular reflects an important aspect of multiway arrays. Indeed, the core idea behind array processing is to include new information in each mode that is not redundant with the information contained in other modes. This will be particularly important when applying tensor decompositions to a given array $\boldsymbol{\mathcal{T}}$.

It is important to understand that although matrices are two-way arrays, they are not considered as multiway arrays since they do not share some crucial properties of multiway arrays that are recalled in what follows. It is however useful to represent a three way array by its slices, *i.e.* matrices obtained by fixing one index of the table. For example, the three way array from Figure 1.1 is efficiently represented by the following stack of matrices :

$$\left[\begin{array}{cc|cc} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{array}\right] \tag{1.1}$$

An easy way to build a three way array is achieved by computing the outer product of three vectors.

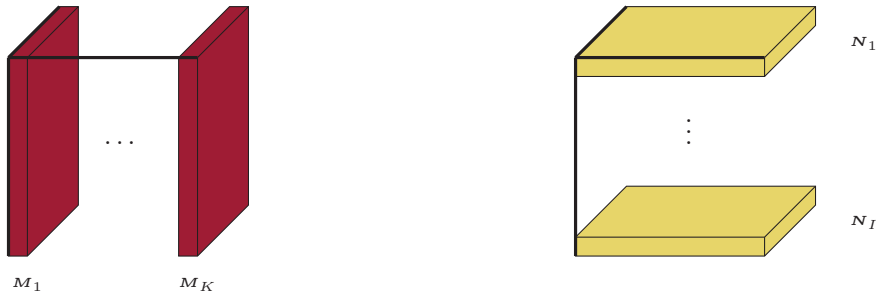
Definition 2 *The outer product $\boldsymbol{\mathcal{T}} = \boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c}$ of three vectors $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ respectively in $\mathbb{R}^K$, $\mathbb{R}^L$ and $\mathbb{R}^M$ is a three way array defined by the following coefficients:*

$$T_{klm} = a_k b_l c_m \tag{1.2}$$

Figure 1.2 provides a graphic explanation of what the outer product does. Since arrays of the same size can be added by summing their coefficients, it is possible to build arrays by sums of outer products of vectors. In fact any array can be described by a sum of outer products if enough outer products are summed. This point will be detailed in length in the section **??**.

A useful tool to manipulate arrays is the reshaping operator. Reshaping an array means transforming the array into another with fewer modes. For three way arrays, reshaping means putting all the coefficients in a matrix or a vector.

First let us have a look at the matricization.



$$\begin{aligned}\boldsymbol{T}_{(1)} &= [\boldsymbol{M}_1|\dots|\boldsymbol{M}_K]\\ \boldsymbol{T}_{(2)} &= [\boldsymbol{N}_1|\dots|\boldsymbol{N}_I]\\ \boldsymbol{T}_{(3)} &= [\boldsymbol{N}_1^T|\dots|\boldsymbol{N}_I^T]\end{aligned}$$

where $\boldsymbol{M}_i \in \mathbb{R}^{K\times M}$ and $\boldsymbol{N}_i \in \mathbb{R}^{L\times M}$

Figure 1.3: Three matricizations $\boldsymbol{T}_{(i)}$ of a three way array $\boldsymbol{\mathcal{T}}$ of size $K \times L \times M$

Definition 3 *A matricization of an array in* $\mathbb{R}^{K\times L\times M}$ *is an isomorphism* ϕ *mapping* $\mathbb{R}^{K\times L\times M}$ *to* $\mathbb{R}^{I\times J}$ *so that for any array* $\boldsymbol{\mathcal{T}}$, $\phi(\boldsymbol{\mathcal{T}})_{ij} = T_{klm}$ *where each couple* (i,j) *is uniquely mapped to one tuple* (k,l,m) *and* $IJ = KLM$.

There are $(KLM)!$ possible matricizations for arrays of size $K \times L \times M$. Of course some of them completely destroy the structure of the data and should not be used in a data mining context. However, preserving structure still leaves some degrees of freedom with respect to choosing a matricization to work with. We choose three matricizations denoted $\boldsymbol{T}_{(i)}$ that preserve the ordering of coefficients for some slices of the cube as presented in Fig 1.3, while preserving the order of multiplications of vectors in folded modes. Indeed, the matricization of a three way array concatenates two modes. For outer products of three vectors, this means transforming the outer product of two of the vectors into a product computing one larger vector. The product corresponding to the outer product after matricization is known as the Kronecker product and extends to matrices:

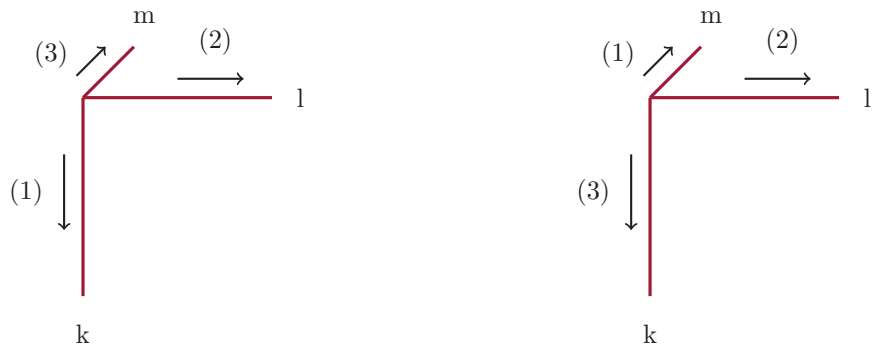
Definition 4 *The Kronecker product of two arrays* $\boldsymbol{A} \in \mathbb{R}^{p_1\times q_1}$ *and* $\boldsymbol{B} \in \mathbb{R}^{p_2\times q_2}$ *is denoted by* $\boldsymbol{A}\boxtimes\boldsymbol{B} \in \mathbb{R}^{p_1p_2\times q_1q_2}$ *and is defined by:*

$$\boldsymbol{A}\boxtimes\boldsymbol{B} := \left[\begin{array}{c|c|c|c} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \dots & a_{1q_1}\boldsymbol{B} \\ \hline a_{21}\boldsymbol{B} & & & \\ \hline \vdots & & & \\ \hline a_{p_11}\boldsymbol{B} & & & a_{p_1q_1}\boldsymbol{B} \end{array}\right]. \tag{1.3}$$

Thus, for matricizations defined in Fig 1.3,

$$\begin{aligned}[\boldsymbol{a}\circ\boldsymbol{b}\circ\boldsymbol{c}]_{(1)} &= \boldsymbol{a}\circ(\boldsymbol{b}\boxtimes\boldsymbol{c})\\ [\boldsymbol{a}\circ\boldsymbol{b}\circ\boldsymbol{c}]_{(2)} &= \boldsymbol{b}\circ(\boldsymbol{a}\boxtimes\boldsymbol{c})\\ [\boldsymbol{a}\circ\boldsymbol{b}\circ\boldsymbol{c}]_{(3)} &= \boldsymbol{c}\circ(\boldsymbol{a}\boxtimes\boldsymbol{b})\end{aligned}. \tag{1.4}$$

This choice of matricization seems natural since outer products, and as we will see later tensor products, are at the core of all multiway array processing. Matricizations are useful when each mode of arrays is put in the first mode of a matricization, but preserving the order of the remaining vectors when moving from arrays to matrices is good practice to clarify computation, especially for arrays with more than three modes.



a) columnwise vectorization (MATLAB) b) suggested vectorization

Figure 1.4: Two possibilities for vectorizing an array. Left is the vectorization suggested in [72] and implemented in MATLAB2014b$^{\text{TM}}$, right is the one used in this manuscript. If T_{klm} is a coefficient of $\boldsymbol{\mathcal{T}}$ of size $K \times L \times M$, then in the vectorized form $\boldsymbol{t}$ it is located at : a) $t_{(m-1)KL+(l-1)K+k}$ b) $t_{(k-1)LM+(l-1)M+m}$

This definition can be exploited to obtain a compact MATLABR2014b$^{\text{TM}}$ code for computing the three unfoldings:

$$\begin{aligned}\boldsymbol{T}_{(1)} &= \text{reshape}\left(\text{permute}\left(\boldsymbol{\mathcal{T}}, [1,3,2]\right), K, L * M\right)\\ \boldsymbol{T}_{(2)} &= \text{reshape}\left(\text{permute}\left(\boldsymbol{\mathcal{T}}, [2,3,1]\right), L, K * M\right)\\ \boldsymbol{T}_{(3)} &= \text{reshape}\left(\text{permute}\left(\boldsymbol{\mathcal{T}}, [3,2,1]\right), M, K * L\right)\end{aligned}$$

.

Another useful way to reshape an array is to turn it into a vector. This operation is called vectorization.

Definition 5 *A vectorization of arrays in $\mathbb{R}^{K\times L\times M}$ is an isomorphism ψ mapping $\mathbb{R}^{K\times L\times M}$ to $\mathbb{R}^N$ so that for any array $\boldsymbol{\mathcal{T}}$, $\psi(\boldsymbol{\mathcal{T}})_i = T_{klm}$ where $i \in [1, N]$ is uniquely mapped to a tuple (k, l, m) and $N = KLM$.*

We denote vec($\boldsymbol{\mathcal{T}}$) the vectorization *b*) of Figure 1.4.

Once again, we choose the only vectorization among many that preserves the order of vectors when vectorizing an outer product, see Figure 1.4. Moreover similarly to matricization, our vectorization of outer products preserves column, row and fiber structure, so that outer products are mapped to Kronecker products :

$$\text{vec}(\boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c}) = \boldsymbol{a} \boxtimes \boldsymbol{b} \boxtimes \boldsymbol{c}. \tag{1.5}$$

This is in contrast with the definition of MATLAB, which does not enjoy this convenient property.

1.1.2 Array manipulation

Now that multiway arrays have been defined, that arrays can be built using the outer product and that we can reshape arrays at will, a natural question to raise is how to manipulate arrays in order to extend fundamental tools of linear algebra, *i.e.* how to

- change the basis of representation of an array ?
- define a scalar product and a norm ?

Euclidian Norm and scalar product

The Euclidian norm of an array can be obtained through its vectorization. For matrices the Euclidian norm of vectors yields the Frobenius norm after vectorization. Similarly, Euclidian norm of vectorization of arrays yields a Frobenius (ℓ_2) norm:

Definition 6 *The Frobenius norm $\|\boldsymbol{\mathcal{T}}\|_F$ of array $\boldsymbol{\mathcal{T}}$ is the non-negative quantity defined by the following equality:*

$$\|\boldsymbol{\mathcal{T}}\|_F^2 = \|vec(\boldsymbol{\mathcal{T}})\|_2^2 = \sum_{klm} T_{klm}^2. \tag{1.6}$$

Defining a scalar product of two tensors of same size follows from this definition using the parallelogram identity. But let us see it differently. To build a scalar product of two arrays $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{\mathcal{T}}'$ corresponding to the canonical scalar product of vectors, we need to "contract" $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{\mathcal{T}}'$ on each mode. A formulation with the trace operator is classically used in quantum physics and is detailed in Appendix A. Here let us see how to build such a scalar product with outer products, and since any array can be defined by a sum of outer products, this scalar product generalizes to all arrays. A scalar product on the space of arrays is the bilinear positive symetric form given by the following:

$$\langle \boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c} \mid \boldsymbol{a}' \circ \boldsymbol{b}' \circ \boldsymbol{c}' \rangle := \langle \boldsymbol{a} \mid \boldsymbol{a}' \rangle \langle \boldsymbol{b} \mid \boldsymbol{b}' \rangle \langle \boldsymbol{c} \mid \boldsymbol{c}' \rangle \tag{1.7}$$

where the scalar products on the right are the Euclidian scalar products in $\mathbb{R}^K$, $\mathbb{R}^L$ and $\mathbb{R}^M$. This definition coincides with the scalar product of matrices and yields the Frobenius norm for arrays defined above. Indeed, let $\boldsymbol{M} = \sum_{r=1}^{R} \boldsymbol{a}_r \circ \boldsymbol{b}_r$ and $\boldsymbol{M}' = \sum_{p=1}^{P} \boldsymbol{a}'_p \circ \boldsymbol{b}'_p$, then

$$\langle \boldsymbol{M} \mid \boldsymbol{M}' \rangle = \mathrm{Tr}(\boldsymbol{M}^\mathsf{T} \boldsymbol{M}') = \sum_{\mathrm{r,p}}^{\mathrm{R,P}} \langle \boldsymbol{a}_\mathrm{r} \mid \boldsymbol{a}'_\mathrm{p} \rangle \langle \boldsymbol{b}_\mathrm{r} \mid \boldsymbol{b}'_\mathrm{p} \rangle. \tag{1.8}$$

The norm and the scalar product described above are invariant with respect to matricization and vectorization of arrays.

Change of basis of representation

To change the basis of representation of an array implies that, first of all, an array is not considered a fixed table of numerical values but rather the expression of an element of $\mathbb{R}^{K \times L \times M}$ in a given basis $\mathcal{B}$. It is shown in the next section that any basis $\mathcal{B}$ of the vector space of finite dimensions arrays can be expressed using a basis for each mode of the array, *i.e.* there exists matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ so that any basis vector in $\mathcal{B}$ is the outer product of vectors in the span of $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$. In other words, to change the basis of representation of an array, it is sufficient to change its coordinates mode-wise.

Actually it makes a lot of sense to operate on a multiway array mode per mode, since each mode carries a different modality with its own scale or measurement unit. To affect an array on only one mode, the N-mode product can be used.

Definition 7 *The 1-mode product of an array $\boldsymbol{\mathcal{T}}$ in $\mathbb{R}^{K \times L \times M}$ with a matrix $\boldsymbol{U}$ in $\mathbb{R}^{K' \times K}$ is denoted by a three way array $\boldsymbol{\mathcal{T}} \bullet_1 \boldsymbol{U}$ in $\mathbb{R}^{K' \times L \times M}$ defined by the following equality:*

$$[\boldsymbol{\mathcal{T}} \bullet_1 \boldsymbol{U}]_{k'lm} = \sum_{k=1}^{K} U_{k'k} T_{klm} \tag{1.9}$$

and can be understood as a contraction of $\boldsymbol{\mathcal{T}}$ with $\boldsymbol{U}$ along the first mode of $\boldsymbol{\mathcal{T}}$ and the second mode of $\boldsymbol{U}$. A similar definition holds for 2-mode $\bullet_2$ and 3-mode $\bullet_3$ products.

There are natural relationships between the N-mode products and unfoldings of a tensor that are recalled in Table 1.1 in section 1.3.1.

Since a change of basis can be computed mode-wise, and since the N-mode product is exactly a mode-wise computation of linear transformation, changing the basis of representation of a three way array goes as follows. Given three orthonormal matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ defining three orthonormal bases, we can express array $\boldsymbol{\mathcal{T}}$ as a new core array $\boldsymbol{\mathcal{G}}$ in the new basis as

$$\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{T}} \bullet_1 \boldsymbol{U}^\mathsf{T} \bullet_2 \boldsymbol{V}^\mathsf{T} \bullet_3 \boldsymbol{W}^\mathsf{T}. \tag{1.10}$$

All the basic computation tools for multiway arrays have now been introduced. Most works on multiway array processing resort only to these definitions and a few formulas that we introduce in the third section and are summarized in Tables 1.1 and 1.2. However, tensors have not yet been discussed. Even though the tensor formalism is not utterly necessary when processing arrays of data, using tensor products allows for a generalization of array processing, along with a deeper understanding of what operation makes sense or not when processing arrays. In other words, the root of array processing lies in understanding multilinear algebra.

1.2 Tensor product formalism

??

In what follows, tensors are introduced as in Schwartz's book [101], which itself is based on works from the group Nicolas Bourbaki [11]. Proofs of all the theorems and properties below can be found in these two references. The goal here is to give insight on what a theoretical definition of tensors is, what are intrinsic properties of tensors, and how they relate to multiway arrays.

1.2.1 Tensor product definition

Given two vector spaces $\mathscr{E}$ and $\mathscr{F}$, it is well known that a third vector space can be built using the cartesian product: $\mathscr{E} \times \mathscr{F} = \{(x, y), x \in \mathscr{E} \text{ and } y \in \mathscr{F}\}$. A difficulty encountered with the cartesian product is when bilinear functions over $\mathscr{E} \times \mathscr{F}$ are to be used. Indeed, suppose one wants to process data contained in a vector of $\mathscr{E} \times \mathscr{F}$, then bilinear operators act mode-wise but linear operators may not. This was actually the main motivation for using tensors, and arrays, in data processing [113].

However, ideally using linear operators instead of bilinear operators would make it possible to use the whole arsenal of results and methods of linear algebra. This is exactly what the tensor product amounts to: it is a bijection from $\mathscr{E} \times \mathscr{F}$ mapping to a vector space where bilinear operators become linear. Below is an exact definition. Proofs of existence and uniqueness up to an isomorphism can be found in [101] p4-5.

Definition 8

- *If $\mathscr{E}$ and $\mathscr{F}$ are vector spaces, there exists a vector space $\mathscr{E} \otimes \mathscr{F}$ and a bilinear mapping $\otimes$ from $\mathscr{E} \times \mathscr{F}$ to $\mathscr{E} \otimes \mathscr{F}$ having the following property:*
  for any bilinear mapping u from $\mathscr{E} \times \mathscr{F}$ to a vector space $\mathscr{G}$, there exists a unique linear mapping v from $\mathscr{E} \otimes \mathscr{F}$ to $\mathscr{G}$ such that $u(x, y) = v(x \otimes y)$ for all $x \in \mathscr{E}$ and $y \in \mathscr{F}$.
- *If $\otimes' : \mathscr{E} \times \mathscr{F} \to \mathscr{E} \otimes' \mathscr{F}$ is another mapping with the same property, there exists a unique linear bijection $w : \mathscr{E} \otimes \mathscr{F} \to \mathscr{E} \otimes' \mathscr{F}$ such that $w \circ \otimes = \otimes'$ and $\otimes = w^{-1} \circ \otimes'$. (uniqueness of the tensor product up to composition with isomorphisms)*

In other words, a tensor product is a canonical bilinear mapping that linearizes the cartesian product $\mathscr{E} \times \mathscr{F}$:

$$\begin{array}{rccc} & \mathscr{E} \times \mathscr{F} & \to & \mathscr{E} \otimes \mathscr{F} \\ \otimes \ : & (x, y) & \mapsto & x \otimes y \end{array} . \tag{1.11}$$

There are two important details in this definition. First, it is not possible to define only $\mathscr{E} \otimes \mathscr{F}$ without adding a definition of the tensor product $\otimes$, and conversely a tensor product needs to be defined as a mapping to a specified space $\mathscr{E} \otimes \mathscr{F}$. In other words, it is necessary to provide the couple $(\mathscr{E} \otimes \mathscr{F}, \otimes)$. Second, there are infinitely many possible tensor products that may be built from two vectors spaces. We will often refer to $\otimes$ as 'the' tensor product, but actually we only choose one instance of $(\mathscr{E} \otimes \mathscr{F}, \otimes)$. This matters to some extent in applications when using matricization and vectorization, as explained further below. However when designing data processing methods on tensors and arrays, a generic tensor product and tensor product space can be choosen without loss of generality, and this simplifies computation compared to sticking to outer products and Kronecker products. Thus in this manuscript, whenever it is possible, the tensor product formalism will be used.

A direct corollary of Definition 8 is that the scalars can be placed arbitrarily in all vectors composing a tensor:

$$\lambda \, (x \otimes y) = \lambda x \otimes y = x \otimes \lambda y \tag{1.12}$$

where scalar λ belongs to the field of construction of $\mathscr{E} \otimes \mathscr{F}$.

1.2.2 Some properties on bases of tensor product spaces

Three fundamental properties are listed below. They are almost mandatory to gain intuition on what is a tensor product space. Linear algebra is one of the fields in mathematics that can be really well understood because most reasoning can be done on bases of vector spaces, and those can be mentally visualized. In tensor algebra, *i.e.* algebra on tensor spaces *i.e.* multilinear algebra, reasoning with bases is also a very good way to get intuition.

Proposition 1 *Let $\{e_i\}$ and $\{f_j\}$ be bases of $\mathscr{E}$ and $\mathscr{F}$, respectively. Then $\{e_i \otimes f_j\}$ is a basis of $\mathscr{E} \otimes \mathscr{F}$ and is called the canonical basis associated with the two given bases. More precisely, let $\otimes$ be a bilinear mapping from $\mathscr{E} \times \mathscr{F}$ to $\mathscr{E} \otimes \mathscr{F}$. $(\mathscr{E} \otimes \mathscr{F}, \otimes)$ defines a tensor product space if and only if there exist two bases $\{e_i\}$ and $\{f_j\}$ such that $\{e_i \otimes f_j\}$ is a basis of $\mathscr{E} \otimes \mathscr{F}$. Then it is true also for any basis of $\mathscr{E}$ and $\mathscr{F}$.*

Given two bases for two vector spaces, building a basis of a third vector space through a bilinear mapping is equivalent to building the tensor product space with this bilinear mapping. This characterization of tensor products can be used in practice to build a tensor product space from two vector spaces. The finite dimension case can be very well understood within this framework as exposed in the next subsection.

However in the infinite dimension case, an interesting example among many is provided in [101].

Example 1 *Let $\mathcal{C}([0,1])$ be the vector space of continuous functions from $[0,1]$ to $\mathbb{R}$. Let $\mathscr{E}$ be some finite dimension normed vector space. One possible instance of $(\mathcal{C}([0,1]) \otimes \mathscr{E}, \otimes)$ is obtained by setting*

$$\phi \otimes e := (\phi\ e : x \mapsto \phi(x)e)\,.$$

This bilinear mapping $\otimes$ can be shown to define a tensor product by proving it maps any basis of $\mathscr{E}$ to a basis of $\mathcal{C}([0,1], \mathscr{E})$, set of continuous mappings from $[0,1]$ to $\mathscr{E}$.

From proposition 1, two important corollaries are easily obtained.

Corollary 1 *If $\mathscr{E}$ and $\mathscr{F}$ have finite dimensions n and m, then $\mathscr{E} \otimes \mathscr{F}$ has finite dimension nm.*

Corollary 2 *The set $\{e \otimes f,\ e \in \mathscr{E}$ and $f \in \mathscr{F}\}$ spans $\mathscr{E} \otimes \mathscr{F}$. Tensors that can be written as $e \otimes f$ are called decomposable, or rank-1 tensors.*

The second corollary is of particular interest in multiway data mining. Indeed, it states that any tensor can be written as a sum of rank-1 tensors. However not all tensors are rank-1, which makes the problem of finding, if it exists, the smallest necessary number of rank-1 tensors to express a given tensor a fascinating problem. This issue is discussed at length in section 1.4.1.

Tensor products can be easily extended to handle more than two vector spaces and all properties and corollaries above hold. When N vector spaces $\mathscr{E}_n$ are used to build a tensor space, elements of this tensor space are said to be tensors of order N.

1.2.3 Finite dimension case: back to arrays

There are many fascinating mathematical problems that can be derived from the discussion above, yet what data scientists are interested in is the processing of multiway arrays. Let us see how computation on multiway arrays can be casted in the tensor product formalism.

It is quite frequent that arrays and tensors are confused. It is possible to hear that arrays are supposedly not true tensors, or sometimes that tensors are arrays but not written in a fixed basis, or some other weird formulation trying to give separate meanings to arrays and tensors. To clarify this point once and for all, an array of size $K \times L \times M$ is simply a vector from $\mathbb{R}^{K \times L \times M}$, and whether or not it is written in a given basis or not has nothing to do with it being a tensor or not. Indeed the same goes for vectors in $\mathbb{R}^n$, they are more than just a table of coordinate but that does not mean an intrinsic formulation of a vector in a vector space is a tensor.

Actually, all arrays are tensors, but not all tensors are arrays. To prove this, it is sufficient to see that $(\mathbb{R}^{K \times L \times M}, \circ')$ is a tensor space built from $\mathbb{R}^K \times \mathbb{R}^L \times \mathbb{R}^M$ with $\circ'$ being the canonical tensor product $\circ'(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) = \boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c}$.

Definition 9 *A third order real array $\boldsymbol{\mathcal{T}}$ of size $K \times L \times M$ is a vector of the tensor space $\left(\mathbb{R}^{K \times L \times M}, \circ'\right)$ where the tensor product is $\circ'(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) = \boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c}$. Thus the space of arrays is $\mathbb{R}^K \circ \mathbb{R}^L \circ \mathbb{R}^M$. This definition is equivalent to definition 1.*

Again $\mathbb{R}^{K\times L\times M}$ is only one instance of a tensor product space $\mathbb{R}^K \otimes \mathbb{R}^L \otimes \mathbb{R}^M$, and the outer product is a particular choice $\otimes := \circ'$ of tensor product among many. Notations are already becoming difficult since we need to differentiate between the outer product $\circ$ (which is the tensor product for two-way arrays) and the tensor product $\circ'$ which is a trilinear function. Similarly, $(\mathbb{R}^{KLM}, \boxtimes')$ is a tensor space where the tensor product is defined using the Kronecker product, $\boxtimes'(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) = \boldsymbol{a} \boxtimes \boldsymbol{b} \boxtimes \boldsymbol{c}$. $\mathbb{R}^{KLM}$ is the same "class" of tensor space than $\mathbb{R}^{K\times L\times M}$ since vectors and three way arrays are related through the reshaping isomorphism.

Another example of arrays casted as tensors is arrays obtained through matricization. Let $\boldsymbol{\mathcal{T}}$ in $\mathbb{R}^{K\times L\times M}$ be a three way array. The matricization on the first mode is an isomorphism. So it defines a tensor space $(\mathbb{R}^{K\times LM}, \otimes)$, but the tensor product is not a simple concatenation of Kronecker product or outer product. If Matricization is defined correctly as in Figure 1.3, then here $\otimes(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) = \boldsymbol{a} \circ \boldsymbol{b} \boxtimes \boldsymbol{c}$. Using the tensor formalism makes things quite clear.

It is possible to further play with names and conventions (and enjoy it all the more). A vector in $\mathbb{R}^I$ where $I = KLM$ may be an element of a tensor product space $(\mathbb{R}^K \boxtimes \mathbb{R}^L \boxtimes \mathbb{R}^M, \boxtimes')$, *i.e.* a third order tensor. So vectors can be tensors of any order, matrices are not only tensor of order 2 and arrays are tensors. Clearly, it is crucial to define what are the initial vector spaces considered to build the tensor space, otherwise the "order" of a tensor is a meaningless concept. Moreover, when designing methods for mining multiway arrays, the fact that an array is a tensor is sufficient. In other words, the Kronecker product, outer product, and any other particular instance of a tensor product is only necessary for numerical computation, but not in theoretical design of methods and algorithms. As a consequence in this manuscript, restricting the discussion to arrays is avoided as much as possible to simplify notations. Matricizations and vectorization will be used when numerical computation of a formula is needed.

1.2.4 Linear operators acting on tensors

Since tensors are vectors of a tensor space, one natural tool to study is linear maps acting on tensors. We have seen that a subspace of linear maps acting on arrays is defined by mode-wise linear operators. In other words, using the N-mode product, a subclass of linear operators can be applied on arrays. An interesting property of tensors stated and proved below, is that whenever dealing with finite dimension tensor spaces, all linear maps can be decomposed into mode-wise linear operators. Which means that the set of mode-wise linear operators is a basis of linear maps on finite dimension tensors, *i.e.* the set of linear maps acting on finite dimension tensors is itself a tensor space.

Proposition 2 *Let $\mathscr{E}$, $\mathscr{E}'$, $\mathscr{F}$ and $\mathscr{F}'$ be four finite vector spaces on a field $\mathcal{K}$, of respective dimensions n, n and m, m. Let $\otimes$ and $\otimes'$ be two tensor products on $\mathscr{E} \times \mathscr{F}$ and $\mathscr{E}' \times \mathscr{F}'$, and consider the following mapping:*

$$\begin{array}{rccc} & \mathcal{L}(\mathscr{E}, \mathscr{E}') \times \mathcal{L}(\mathscr{F}, \mathscr{F}') & \rightarrow & \mathcal{L}(\mathscr{E} \otimes \mathscr{F}, \mathscr{E}' \otimes' \mathscr{F}') \\ \otimes_p : & (u, v) & \mapsto & u \otimes_p v : (x \otimes y) \mapsto u(x) \otimes' v(y) \end{array}$$

where $\mathcal{L}(\mathscr{E}, \mathscr{E}')$ is the linear space of linear operators mapping $\mathscr{E}$ to $\mathscr{E}'$. $\mathcal{L}(\mathscr{E} \otimes \mathscr{F}, \mathscr{E}' \otimes' \mathscr{F}')$ associated with the bilinear mapping $\otimes_p$ is a tensor space, i.e.

$$\mathcal{L}(\mathscr{E} \otimes \mathscr{F}, \mathscr{E}' \otimes' \mathscr{F}') = \mathcal{L}(\mathscr{E}, \mathscr{E}') \otimes_p \mathcal{L}(\mathscr{F}, \mathscr{F}').$$

A short proof of this proposition can be found pages 72-73 in [57]. Here a simple proof using tools presented above is derived. To prove proposition 2, we only need to check that $\otimes_p$ maps one basis of $\mathcal{L}(\mathscr{E}, \mathscr{E}') \times \mathcal{L}(\mathscr{F}, \mathscr{F}')$ to a free family of $\mathcal{L}(\mathscr{E} \otimes \mathscr{F}, \mathscr{E}' \otimes' \mathscr{F}')$, which yields injectivity. Then we will be able to conclude arguing that these two spaces have the same dimension, so that by Green's theorem, $\otimes_p$ is a bijective bilinear map from the cartesian product space to a linear space, which is exactly what a tensor product is.

Let $\{e_i\}$, $\{e'_i\}$, $\{f_j\}$ and $\{f'_j\}$ be some bases of $\mathscr{E}, \mathscr{E}', \mathscr{F}$ and $\mathscr{F}'$. Define the following basis for $\mathcal{L}(\mathscr{E}, \mathscr{E}')$:

$$u_i(e_j) = \delta_{ij} e'_i$$

where δ_{ij} is the Kronecker symbol equal to 1 if and only if i equals j. Define the basis $\{v_i\}$ similarly. Let us prove that $\{u_i \otimes_p v_j\}$ is a free family.

Given some $\{\lambda_{ij}\}$, suppose for any $x \otimes y$ in $\mathscr{E} \otimes \mathscr{F}$

$$\sum_{i,j} \lambda_{ij} u_i(x) \otimes v_j(y) = 0.$$

This is equivalent to

$$\sum_{i,j} \lambda_{ij} \nu_i \nu'_j e'_i \otimes f'_j = 0$$

where ν_i and ν'_j are coefficients of x and y in bases $\{e_i\}$ and $\{f_j\}$. Since this is true for any x, y and that $\{e'_i \otimes' f'_j\}$ is a basis of $\mathscr{E}' \otimes' \mathscr{F}'$, all lambdas have to go to zero, thus proving proposition 2. ▫

Of course this can be extended to cover non-square linear operators. Proposition 2 means that the set of linear operators on finite dimension tensors is itself a tensor space $\mathcal{L}(\mathscr{E}, \mathscr{E}') \otimes_p \mathcal{L}(\mathscr{F}, \mathscr{F}')$. A useful class of linear maps on tensors is the set of rank-1 linear operators $\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W}$ (in the whole manuscript the notation $\otimes_p$ is simplified to $\otimes$ since there is no ambiguity with the tensor product of the initial tensor space). They are exactly linear operators that act mode-wise on each mode of an array. That is, the action of $\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W}$ on a tensor $\boldsymbol{\mathcal{T}}$ can be computed in the space of arrays through the N-mode product:

$$(\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\,\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{T}} \bullet_1 \boldsymbol{U} \bullet_2 \boldsymbol{V} \bullet_3 \boldsymbol{W}. \tag{1.13}$$

The notation $(\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\,\boldsymbol{\mathcal{T}}$ is naturally inherited from linear algebra notations $\boldsymbol{U}\boldsymbol{x}$ of the application of linear operator $\boldsymbol{U}$ on vector $\boldsymbol{x}$.

1.3 Computation rules for finite-dimension tensors

1.3.1 Computation rules

Manipulation of array notations by hand are known to be cumbersome, since computation rules are supposedly non-intuitive. This is particularly true when using outer and Kronecker products. However computation is natural when using the tensor product formalism. Proposition 2 paves the way for composition of linear operators with the following rules:

$$\begin{array}{rcl} (\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\,(\boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c}) & = & \boldsymbol{U}\boldsymbol{a} \otimes \boldsymbol{V}\boldsymbol{b} \otimes \boldsymbol{W}\boldsymbol{c} \\ (\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\,(\boldsymbol{U}' \otimes \boldsymbol{V}' \otimes \boldsymbol{W}') & = & \boldsymbol{U}\boldsymbol{U}' \otimes \boldsymbol{V}\boldsymbol{V}' \otimes \boldsymbol{W}\boldsymbol{W}' \end{array}. \tag{1.14}$$

The first rule comes from the definition of the tensor product among linear operators. The second rule is simply the composition of separable linear operators, and is also a direct corollary of their definition sometimes coined as the mixed product rule. If the chosen tensor product for the tensor space is the outer product, then the tensor product for operators is not a usual product between matrices, and the generic symbol $\otimes$ is used throughout the manuscript. Moreover, the outer product does not need to be written with a specific symbol $\circ$ since using a generic symbol $\otimes$ for vectors does not lead to any ambiguity. However, for the Kronecker product, we stick with the symbol $\boxtimes$ since the distinction between a tensor product in $\mathbb{R}^K \otimes \mathbb{R}^L \otimes \mathbb{R}^M$ and its isomorphic Kronecker product is necessary when dealing with reshaping.

Most computation rules that are used in multiway array processing are summed up in tables 1.1 and 1.2. Since these rules are quite trivial but proving them all one by one is cumbersome, we let the reader prove and understand each of them himself. Let us give an example of a typical computation with the tensor product formalism.

Example 2 *Let $\boldsymbol{a}_r$, $\boldsymbol{b}_r$ and $\boldsymbol{c}_r$ be real vectors of finite dimension, $r \leqslant R$ for some integer R. Let $\boldsymbol{1}_r$ be the vector of zeros having only the r^{th} coefficient equals to 1. Then the following holds:*

$$\begin{aligned} \sum_{r=1}^{R} \boldsymbol{a}_r \otimes \boldsymbol{b}_r \otimes \boldsymbol{c}_r &= \sum_{r=1}^{R} \boldsymbol{A}\boldsymbol{1}_r \otimes \boldsymbol{B}\boldsymbol{1}_r \otimes \boldsymbol{C}\boldsymbol{1}_r \\ &= (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C}) \sum_{r=1}^{R} \boldsymbol{1}_r \otimes \boldsymbol{1}_r \otimes \boldsymbol{1}_r \\ &= (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\,\boldsymbol{\mathcal{I}}_R \end{aligned} \tag{1.34}$$

$$\operatorname{vec}\left(\bigotimes_{i=1}^{N} \boldsymbol{a}^{(i)}\right) = \boxtimes_{i=1}^{N} \boldsymbol{a}^{(i)} \text{ in the same order} \tag{1.15}$$

$$\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}^{T} = (\boldsymbol{A} \otimes \boldsymbol{B})\,\boldsymbol{X} \tag{1.16}$$

$$\operatorname{vec}\left(\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}^{T}\right) = (\boldsymbol{A} \boxtimes \boldsymbol{B})\operatorname{vec}(\boldsymbol{X}) \tag{1.17}$$

$$\operatorname{vec}\left(\left(\bigotimes_{i=1}^{N} \boldsymbol{U}_i\right)\boldsymbol{\mathcal{Y}}\right) = \left(\boxtimes_{i=1}^{N} \boldsymbol{U}_i\right)\operatorname{vec}(\boldsymbol{\mathcal{Y}}) \tag{1.18}$$

$$\boldsymbol{A}\boldsymbol{X} + \boldsymbol{X}\boldsymbol{B} = \boldsymbol{D} \Leftrightarrow \left((\boldsymbol{A} \boxtimes \boldsymbol{I}) + \left(\boldsymbol{I} \boxtimes \boldsymbol{B}^{\mathsf{T}}\right)\right)\operatorname{vec}(\boldsymbol{X}) = \operatorname{vec}(\boldsymbol{D}) \tag{1.19}$$

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \bullet_i \boldsymbol{U} \Leftrightarrow \boldsymbol{Y}_{(i)} = \boldsymbol{U}\boldsymbol{X}_{(i)} \tag{1.20}$$

$$\left[\left(\bigotimes_{j=1}^{N} \boldsymbol{U}_j\right)\boldsymbol{\mathcal{Y}}\right]_{(i)} = \boldsymbol{U}_i \boldsymbol{Y}_{(i)} \left(\boxtimes_{j\neq i}^{N} \boldsymbol{U}_j^{T}\right) \tag{1.21}$$

$$\boldsymbol{\mathcal{Y}} = \sum_{r=1}^{R} \bigotimes_{j=1}^{N} \boldsymbol{a}_r^{(j)} \Leftrightarrow \boldsymbol{Y}_{(i)} = \sum_{r=1}^{R} \boldsymbol{a}_r^{(i)} \otimes \boxtimes_{j\neq i}^{N} \boldsymbol{a}_r^{(i)} \tag{1.22}$$

Table 1.1: Some useful properties of reshaping operators

$$(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{C} \otimes \boldsymbol{D}) = (\boldsymbol{A}\boldsymbol{C} \otimes \boldsymbol{B}\boldsymbol{D}), \text{ if compatible} \tag{1.23}$$

$$(\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})^{-1} = \left(\boldsymbol{U}^{-1} \otimes \boldsymbol{V}^{-1} \otimes \boldsymbol{W}^{-1}\right) \tag{1.24}$$

$$\left(\bigotimes_{i=1}^{N} \boldsymbol{U}_i\right)\left(\bigotimes_{i=1}^{N} \boldsymbol{a}^{(i)}\right) = \left(\bigotimes_{i=1}^{N} \boldsymbol{U}_i \boldsymbol{a}^{(i)}\right) \tag{1.25}$$

$$(\boldsymbol{A} \boxtimes \boldsymbol{B})(\boldsymbol{C} \odot \boldsymbol{D}) = (\boldsymbol{A}\boldsymbol{C} \odot \boldsymbol{B}\boldsymbol{D}), \text{ if compatible} \tag{1.26}$$

$$\operatorname{Tr}_1(\boldsymbol{A} \otimes \boldsymbol{B}) = \operatorname{Tr}(\boldsymbol{A})\,\boldsymbol{B};\ \operatorname{Tr}_2(\boldsymbol{A} \otimes \boldsymbol{B}) = \boldsymbol{A}\operatorname{Tr}(\boldsymbol{B}) \tag{1.27}$$

$$(\boldsymbol{A} \otimes \boldsymbol{B})^{T} = \boldsymbol{A}^{T} \otimes \boldsymbol{B}^{T} \tag{1.28}$$

$$\boldsymbol{I}_{(k)}(\boldsymbol{A} \boxtimes \boldsymbol{B} \boxtimes \boldsymbol{C}) = \boldsymbol{A} \odot \boldsymbol{B} \odot \boldsymbol{C} \tag{1.29}$$

$$\sum_{r=1}^{R} \boldsymbol{a}_r \boxtimes \boldsymbol{b}_r = (\boldsymbol{A} \odot \boldsymbol{B})\,\mathbb{1}, \text{ where } \boldsymbol{A} = [\boldsymbol{a}_1 \ldots \boldsymbol{a}_R] \tag{1.30}$$

$$\sum_{r=1}^{R} \boldsymbol{a}_r^{(i)} \otimes \boxtimes_{j\neq i}^{N} \boldsymbol{a}_r^{(j)} = \boldsymbol{A}_i \boldsymbol{I}_{(i)} \left(\boxtimes_{j\neq i}^{N} \boldsymbol{A}_j^{T}\right) = \boldsymbol{A}_i \left(\bigodot_{j\neq i}^{N} \boldsymbol{A}_j^{T}\right) \tag{1.31}$$

$$\frac{\partial \bigotimes_{j=1}^{N} \boldsymbol{a}_j}{\partial \boldsymbol{a}_i} = \boldsymbol{a}_1 \otimes \ldots \otimes \boldsymbol{a}_{i-1} \otimes \boldsymbol{I} \otimes \ldots \otimes \boldsymbol{a}_N \tag{1.32}$$

$$\left|\bigotimes_{i=1}^{N} \boldsymbol{U}_i\right| = \prod_{i=1}^{N} |\boldsymbol{U}_i|^{\prod_{j\neq i}^{N} n_j} \text{ where } |\boldsymbol{U}| = \det(\boldsymbol{U}) \tag{1.33}$$

Table 1.2: Some useful relations

where $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_r]$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are defined similarly, and $\boldsymbol{\mathcal{I}}_R$ is a tensor of size $R \times R \times R$ with ones on the diagonal in the canonical basis.

Definition 10 (Khatri-Rao product) *Given two arrays $\boldsymbol{A}$ and $\boldsymbol{B}$ in respectively $\mathbb{R}^{I \times J}$ and $\mathbb{R}^{I' \times J}$, the Khatri-Rao product of $\boldsymbol{A}$ and $\boldsymbol{B}$ is an array $\boldsymbol{A} \odot \boldsymbol{B}$ in $\mathbb{R}^{II' \times J}$ obtained by columnwise Kronecker products:*

$$\boldsymbol{A} \odot \boldsymbol{B} = \left[\boldsymbol{A}_{:1} \boxtimes \boldsymbol{B}_{:1}, \ldots, \boldsymbol{A}_{:p} \boxtimes \boldsymbol{B}_{:p}\right]. \tag{1.35}$$

The Khatri-Rao product appears along with matricization operators applied on sums of decomposable tensors. Indeed, the product of matrix $\boldsymbol{M}$ in $\mathbb{R}^{R \times N}$ applied on the right to a matricized diagonal tensor of ones $\boldsymbol{\mathcal{I}}$ selects only rows of $\boldsymbol{M}$ with indices in $\{R(r-1)+r, r \in [\![1, R]\!]\}$. It follows that for $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ respectively in $\mathbb{R}^{R \times K}$,$\mathbb{R}^{R \times L}$ and $\mathbb{R}^{R \times M}$ and for all k,

$$\boldsymbol{I}_{(k)} \left(\boldsymbol{A} \boxtimes \boldsymbol{B} \boxtimes \boldsymbol{C}\right) = \boldsymbol{A} \odot \boldsymbol{B} \odot \boldsymbol{C}. \tag{1.36}$$

Below are some further notions that need to be introduced for later.

1.3.2 Trace operators and derivatives

When developing algorithms to mine data stored in multiway arrays, the notion of derivative of multilinear maps on arrays is necessary. Usually arrays are matricized to obtain a linear algebra formalism, but this is not mandatory. Using partial traces, it is possible to compute gradients of norms of transformed arrays without resorting to reshaping.

Definition 11 (Partial Trace) *The partial trace $\mathrm{Tr}_{\mathscr{E}}$ over $\mathscr{E}$ of tensors in $\mathscr{E} \otimes \mathscr{F}$ is the operator that collapses the tensor on the mode corresponding to $\mathscr{E}$, but leaving the other modes intact. Formally, $\mathrm{Tr}_{\mathscr{E}}$ is the unique linear operator $\mathrm{Tr}_{\mathscr{E}} : \mathcal{L}(\mathscr{E} \otimes \mathscr{F}, \mathscr{E} \otimes \mathscr{F}) \to \mathcal{L}(\mathscr{F}, \mathscr{F})$ so that $\mathrm{Tr}_{\mathscr{E}}(\boldsymbol{U} \otimes \boldsymbol{V}) = \mathrm{Tr}\,(\boldsymbol{U})\,\boldsymbol{V}$. Alternatively, if $\mathscr{E}$ is implicit, $\mathrm{Tr}_{\mathscr{E}}$ is denoted by Tr_1.*

A discussion on the trace and partial trace operators, the Frobenius norm and a scalar product of tensors is provided in Appendix A. It is omitted here for simplicity since the definitions of Frobenius norm and scalar product given for arrays are enough for processing arrays.

The next proposition provides an alternative way to compute the gradient of the cost function related to tensor decompositions introduced in the next section. The proof of this result can be found in Appendix A.

Proposition 3 (Gradient of the Frobenius norm of a linearly transformed tensor) *Let $\gamma(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}) = \|(\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\,\boldsymbol{\mathcal{T}}\|_F$. Then with $\mathrm{Tr}_{2,3}$ being the partial trace over the second and third vector space, the gradient of γ with respect to $\boldsymbol{U}$ is computed as follows:*

$$\frac{\partial \gamma}{\partial \boldsymbol{U}} = \boldsymbol{U}\mathrm{Tr}_{2,3}\left((\boldsymbol{I} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\,\boldsymbol{\mathcal{T}}\,((\boldsymbol{I} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\,\boldsymbol{\mathcal{T}})^*\right) \tag{1.37}$$

where $\boldsymbol{\mathcal{T}}^*$ is the linear form associated with vector $\boldsymbol{\mathcal{T}}$. The gradient for other variables can be obtained by circular permutation. Another form of this gradient which is often found in the literature goes as follows:

$$\frac{\partial \gamma}{\partial \boldsymbol{U}} = \boldsymbol{U}\boldsymbol{T}_{(1)}\left(\boldsymbol{V}^{\mathsf{T}}\boldsymbol{V} \boxtimes \boldsymbol{W}^{\mathsf{T}}\boldsymbol{W}\right)\boldsymbol{T}_{(1)}^{\mathsf{T}}. \tag{1.38}$$

1.3.3 Array normal distribution

Finally, let us define a probability suitable for describing Gaussian noise for arrays. Once again, the usual multivariate normal distribution can be used after applying the vectorization operator, but array normal distributions are directly applicable when dealing with noisy arrays.

Definition 12 (Array normal distribution) *Let $\boldsymbol{\mathcal{X}}$ be a multivariate random variable in $\mathbb{R}^{n_1 \times \cdots \times n_N}$. $\boldsymbol{\mathcal{X}}$ follows an array normal distribution of mean $\boldsymbol{\mathcal{M}}$ and with tensor covariance $\boldsymbol{\Gamma} = \bigotimes_{i=1}^{N} \boldsymbol{\Sigma}_i$ if and only if*

$$p\left(\boldsymbol{\mathcal{X}} | \boldsymbol{\mathcal{M}}, \boldsymbol{\Gamma}\right) = \frac{exp\left(-\frac{\|\boldsymbol{\Gamma}^{-\frac{1}{2}}(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{M}})\|^2}{2}\right)}{(2\pi)^{\prod_i \frac{n_i}{2}} |\boldsymbol{\Gamma}|^{\frac{1}{2}}}$$

where $\boldsymbol{\Gamma}^{-\frac{1}{2}} = \bigotimes_{i=1}^{N} \boldsymbol{\Sigma}_i^{-\frac{1}{2}}$ and $\boldsymbol{\Sigma}_i$ are symmetric. It is denoted as $\boldsymbol{\mathcal{X}} \sim \mathcal{AN}\left(\boldsymbol{\mathcal{M}}, \boldsymbol{\Gamma}\right)$

This definition implicitly requires that the covariance is invertible, an assumption which is also necessary for multivariate Gaussian distributions to obtain a density function.

The main advantage of array normal laws is that the covariance is mode-wise. For example, suppose an array $\boldsymbol{\mathcal{T}}$ is built by drawing each coefficient from identical identically distributed (i.i.d.) centered Gaussian distributions. Then $\boldsymbol{\mathcal{T}}$ should follow the simplest array normal law, with zero mean and diagonal covariance. If $\boldsymbol{\mathcal{T}}$ is linearly modified mode-wise, then similarly to regular multivariate Gaussian distributions, the covariance is multiplied on the left and right by the same linear transformation, but only on the modified mode. In other words, if $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ are full row-rank matrices and $\boldsymbol{\mathcal{T}}$ follows an array normal law of mean $\boldsymbol{\mathcal{M}}$ and diagonal unit covariance,

$$(\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\,\boldsymbol{\mathcal{T}} \sim \mathcal{AN}\left((\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\,\boldsymbol{\mathcal{M}}, \boldsymbol{U}\boldsymbol{U}^{\mathsf{T}} \otimes \boldsymbol{V}\boldsymbol{V}^{\mathsf{T}} \otimes \boldsymbol{W}\boldsymbol{W}^{\mathsf{T}}\right). \tag{1.39}$$

1.4 Tensor decomposition models

There is a subtle difficulty when learning about multiway data processing. It is often believed that once data has been shaped into a multiway array, tensor decomposition techniques will work. This false belief probably takes roots in the fact that hypotheses to be checked by multiway data to be efficiently mined by tensor decomposition models are fuzzy. Actually these hypotheses depend on which decomposition is to be applied, and casting these decomposition models in the tensor algebra language sheds light on these hypotheses. Moreover, designing tensor decomposition models accounting for some prior knowledge on the data to be processed is a crucial research topic in both multiway array and signal processing.

In what follows, the two mostly used tensor decomposition models in data mining are introduced. They have proven extremely efficient for environmental data mining [72], but are also the simplest models that can be thought of and will be used as ingredients for models explored in the second part of this manuscript.

1.4.1 Canonical Polyadic decomposition

Given a tensor $\boldsymbol{\mathcal{T}}$ in $\mathbb{R}^K \otimes \mathbb{R}^L \otimes \mathbb{R}^M$, what is the smallest number of 'simple' tensors, *i.e.* rank-1 tensors, that add up to *exactly* $\boldsymbol{\mathcal{T}}$, and what are their coordinates? Solving this question for $\boldsymbol{\mathcal{T}}$ along with subsequent theoretical problems means finding the expression of the data in $\boldsymbol{\mathcal{T}}$ as a sum of simple components. In other words, it leads to identify the underlying structure of the data. This decomposition is called the Canonical Polyadic Decomposition of $\boldsymbol{\mathcal{T}}$. The CPD was first studied for decomposing polynomials and was known as the Waring problem, while the idea of decomposing tensors is due to Hitchcock [64] and was later rediscovered in psychometrics [25,113]. Nowadays literature on the CPD is quite extensive.

Definition 13 *Let $\boldsymbol{\mathcal{T}}$ in $\mathbb{R}^K \otimes \mathbb{R}^L \otimes \mathbb{R}^M$. A Canonical Polyadic Decomposition or PARAFAC decomposition of $\boldsymbol{\mathcal{T}}$ of rank R is defined as follows:*

$$\boldsymbol{\mathcal{T}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\,\boldsymbol{\Sigma}_R = \sum_{r=1}^{R} \sigma_r \boldsymbol{a}_r \otimes \boldsymbol{b}_r \otimes \boldsymbol{c}_r = \sum_{r=1}^{R} \sigma_r \boldsymbol{\mathcal{T}}_r \tag{1.40}$$

where $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ respectively in $\mathbb{R}^{K\times R}$, $\mathbb{R}^{L\times R}$ and $\mathbb{R}^{M\times R}$ are called the (loading) factors of $\boldsymbol{\mathcal{T}}$, $\boldsymbol{\Sigma}_R$ is a diagonal tensor in $\mathbb{R}^R \otimes \mathbb{R}^R \otimes \mathbb{R}^R$ containing the σ_r, and $\boldsymbol{\mathcal{T}}_r$ are decomposable rank one tensors in $\mathbb{R}^K \otimes \mathbb{R}^L \otimes \mathbb{R}^M$.

The rank of a tensor $\text{rank}(\boldsymbol{\mathcal{T}})$ is the smallest R needed to exactly write $\boldsymbol{\mathcal{T}}$ as a CPD of rank R. It coincides with matrix rank for two-way arrays. Actually since matrix CPD is an interesting degenerate instance of the CPD, let us explore it first.

Matrix case

Let $\boldsymbol{M} \in \mathbb{R}^{n\times m}$ be a matrix of rank R. It is usually understood that columns and rows of $\boldsymbol{M}$ span two vector spaces of dimension R, thus $R \leqslant \min(n, m)$. Then by choosing some basis $\boldsymbol{U}$ for the column space with full column rank, since each column of $\boldsymbol{M}$ is in the span of $\boldsymbol{U}$, there is some

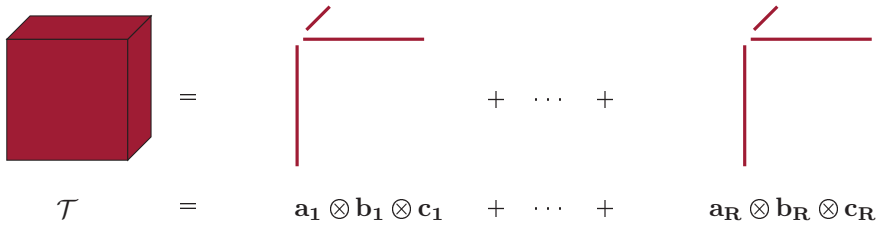


Figure 1.5: Representation of the CPD of $\boldsymbol{\mathcal{T}}$ as a decomposition into a sum of rank-1 tensors

matrix $\boldsymbol{N}$ so that $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{N}^{\mathsf{T}}$. Since $\boldsymbol{M}$ has column rank R and $\boldsymbol{U}$ is full column rank, $\boldsymbol{N}$ is also full column rank. Moreover, it can be checked that

$$\boldsymbol{U}\boldsymbol{N}^{\mathsf{T}} = \sum_{r=1}^{R} \boldsymbol{u}_r \circ \boldsymbol{n}_r \tag{1.41}$$

where $\boldsymbol{u}_r$ and $\boldsymbol{n}_r$ are the columns of $\boldsymbol{U}$ and $\boldsymbol{N}$. Since there cannot be a sum with fewer outer products of linear combinations of $\boldsymbol{u}_r$ and $\boldsymbol{n}_r$ equal to $\boldsymbol{M}$, otherwise $\boldsymbol{U}$ and $\boldsymbol{N}$ would not be full column rank, the right hand side of (1.41) is a CPD of $\boldsymbol{M}$ of rank R. Conversly, a matrix $\boldsymbol{U}\boldsymbol{N}^{\mathsf{T}}$ where each factor is full column rank is clearly a rank R matrix. This proves that the matrix rank and tensor rank coincide for two-way arrays.

Now for a matrix of rank R, clearly there are multiple possible CPD's. In fact for any invertible matrix $\boldsymbol{P}$,

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{N}^{\mathsf{T}} = \boldsymbol{U}\boldsymbol{P}\boldsymbol{P}^{-1}\boldsymbol{N}^{T} = \left(\boldsymbol{U}\boldsymbol{P} \otimes \boldsymbol{N}\boldsymbol{P}^{-\mathsf{T}}\right) \boldsymbol{\mathcal{I}}_R. \tag{1.42}$$

This means that there is not a unique set of factors yielding a CPD of $\boldsymbol{M}$. That is why additional constraints are often imposed in matrix decompositions. Indeed, in Principal Component Analysis (PCA), orthogonality constraints are imposed on the factors [95], while in Nonnegative Matrix Factorization, the factors are set to be non-negative [79].

In the next paragraph, we study existing results on uniqueness of the CPD for tensors of order higher than or equal to three.

Identifiability of the CPD

A major question to be asked about the CPD of tensors of order three or more is, given a tensor of rank R, can components in its CPD be uniquely identified ? After a quick look at (1.40), it can be seen that the norms of vectors $\boldsymbol{a}_r$, $\boldsymbol{b}_r$ and $\boldsymbol{c}_r$ may all be pulled into σ_r without modifying the result. Thus there is a trivial identifiability issue with the norms of the columns of factors that is called scaling ambiguity:

$$\sum_{r=1}^{R} \boldsymbol{a}_r \otimes \boldsymbol{b}_r \otimes \boldsymbol{c}_r = \sum_{r=1}^{R} \sigma_r \boldsymbol{a}_r \otimes \frac{\boldsymbol{b}_r}{\sigma_r} \otimes \boldsymbol{c}_r. \tag{1.43}$$

It can be dealt with in practice by either pulling all the norms in σ_r and normalizing all columns of factors, or by pulling the norm of columns of two factors in the third factor and setting $\sigma_r = 1$. Note that the scaling indeterminacy is in fact a parametrization ambiguity since it appears when the rank one tensors in the CPD are expressed as the tensor product of vectors.

Another source of loss of identifiability of the CPD is the Kruskal rank deficiency of factor matrices. Indeed, if two rank one components have the same span in one mode, then vectors in the two other modes are identifiable only up to a rotation. This fact is detailed in section 3.1.2 dealing with the PARALIND model.

Apart from these observations, a first result is due to Kruskal [77], who proved that should the number of components R be small enough and without too much collinearity between columns of factors, components can always be identified from the data, with restrictions stated above. The bound on tensor rank is proven in an understandable manner in [109]. Moreover, work has been conducted that provides bounds beyond Kruskal's [**?**, 49, 50]. However, a quick computation of the number of parameters and equations in 1.40 shows that there are at most $(K + L + M - 2)R$

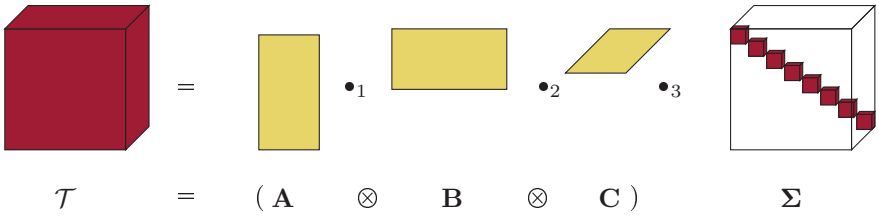


Figure 1.6: Representation of the CPD of $\boldsymbol{\mathcal{T}}$ as a change of representation basis

independent parameters and KLM equations. A naive bound on the number of components R is thus given by

$$R \leqslant \lfloor \frac{KLM}{(K+L+M-2)} \rfloor \tag{1.44}$$

which happens to be a generic upper-bound on the rank to ensure identifiability of components in most cases called non-defective [27]. The minus two parameters comes from the scaling ambiguity discussed above.

What should be remembered in practice is that a tensor $\boldsymbol{\mathcal{T}}$ satisfying the strict inequality (1.44) admits a unique set of factors $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ in its CPD with probability one. In data mining applications, this assumption is often - but not always - verified since $\lfloor \frac{KLM}{(K+L+M-2)} \rfloor$ is usually very large and a low rank assumption is made on the noiseless data, as explained below.

Interpretations of the CPD

There are several interpretations of the CPD when applied on multiway data. First, since the CPD is unique in most applications, it can be used to infer parameters of an underlying multilinear model without adding further constraints, see Figure 1.5. Therefore, CPD is an instance of the Blind Source Separation problem [38]. In BSS, observed data are the result of mixing output of a few sources where neither the sources nor the mixing processes are known. However if this mixing can be shown to be multilinear, then the CPD of data measurements allows to recover exactly the underlying sources and mixing.

Another interpretation of CPD is diagonalization of tensors or dimensionality reduction, see Figure 1.6. CPD expresses the tensor in a new basis spanning the subspace where data are embedded, and this basis of a multilinear subspace is usually not orthogonal. In other words, the bases of observation of each mode are not the ones in which the data are expressed as simply as possible, and the CPD finds these bases of simple representation for all modes simultaneously. Therefore, the CPD answers questions about the relationship between all modes of an array, and is not simply a mode-wise treatment of the data. This last remark was actually the starting motivation for using CPD in data mining and remains the main asset of CPD.

Approximate CPD

Up until now, only an exact CPD of multiway data has been discussed. However, it is more realistic to assume that data would be corrupted by noise, or that a multilinear model would not exactly fit the data. Moreover, since the rank of a CPD refers to a number of meaningful components contained in the data, it is usually supposed small with respect to the dimensions of the data array. Thus, exact decomposition of multiway arrays (1.40) may not be sufficient in many applications.

More precisely, to apply CPD to real-life problems, there are three hypotheses to be verified.

1. The multiway array of data $\boldsymbol{\mathcal{T}}$ is corrupted by additive noise, so that there exist a data array $\boldsymbol{\mathcal{X}}$ and a random variable $\boldsymbol{\mathcal{E}}$ following some probability density so that

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{E}}. \tag{1.45}$$

2. The data array of interest $\boldsymbol{\mathcal{X}}$ is generated by a small number of components of interest and by a multilinear process. Therefore, $\boldsymbol{\mathcal{X}}$ admits an exact CPD of rank R which can be considered small with respect to the sizes of the data array:

$$\boldsymbol{\mathcal{X}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\, \boldsymbol{\Sigma}_R. \tag{1.46}$$

   This hypothesis is strong since it will not be verified by noiseless data with a small number of underlying components expressed in a non-multilinear model or in a model poorly approximated by a multilinear model. The decomposition of $\boldsymbol{\mathcal{X}}$ also needs to be identifiable.

3. The noise $\boldsymbol{\mathcal{E}}$ has a high rank, that is, the noise does not have some inherent multilinear structure. Since all tensors admit a CPD, this condition does not state that $\boldsymbol{\mathcal{E}}$ does not admit a CPD, but it does imply that the rank of the CPD of $\boldsymbol{\mathcal{E}}$ is as high as expected of a randomly drawn tensor, *i.e.* $\lfloor \frac{KLM}{(K+L+M-2)} \rfloor + 1$ [1].

Some examples of applications where all hypothesis are verified can be found in chapter 2.

In many cases the noise $\boldsymbol{\mathcal{E}}$ follows an array normal distribution with zero mean and diagonal tensor covariance $\sigma_n^2 \boldsymbol{I}^{\otimes 3}$. Then hypothesis 3 is met, and under 1 and 2, equation (1.45) yields the following distribution on the data:

$$\boldsymbol{\mathcal{T}} \sim \mathcal{AN}\left((\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\, \boldsymbol{\Sigma}_R, \sigma_n^2 \boldsymbol{I}^{\otimes 3}\right). \tag{1.47}$$

Even though the existence of $\boldsymbol{\mathcal{X}}$ is ensured by the model, the existence of the best rank R approximation $\hat{\boldsymbol{\mathcal{X}}}$ of $\boldsymbol{\mathcal{T}}$ is not guarantied and this poses serious problems in applications. Indeed the set of rank R tensors is not closed, and the set of tensors that do not admit a best low-rank approximate is not negligible [47]. In this context additional constraints are needed on the decomposition so that the intersection of the set of rank R tensors with the constraint space is closed [83]. This topic is covered in chapter 3.

The next paragraph explains how to recover the factors of $\boldsymbol{\mathcal{X}}$ knowing model (1.47).

CPD algorithms with Gaussian noise

Let us derive a cost function to be minimized for solving the approximate CPD. The maximum log-likelihood estimate of $\boldsymbol{\mathcal{X}}$ is easily obtained :

$$\begin{array}{rcl} \hat{\boldsymbol{\mathcal{X}}} & = & \underset{\text{s.t. } \boldsymbol{\mathcal{X}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\, \boldsymbol{\Sigma}_R}{\operatorname{argmin}} \|\boldsymbol{\mathcal{T}} - \boldsymbol{\mathcal{X}}\|_F^2 \end{array}. \tag{1.48}$$

Problem (1.48) can be seen as a linear system $\operatorname{vec}(\boldsymbol{\mathcal{T}}) = \boldsymbol{M} \operatorname{vec}(\boldsymbol{\mathcal{I}}_R)$ under the constraint $\boldsymbol{M} = \boldsymbol{A} \boxtimes \boldsymbol{B} \boxtimes \boldsymbol{C}$:

$$\begin{array}{rcl} \hat{\boldsymbol{\mathcal{X}}} & = & \underset{\text{s.t. } \boldsymbol{\mathcal{M}} = \boldsymbol{A} \boxtimes \boldsymbol{B} \boxtimes \boldsymbol{C}}{\operatorname{argmin}} \|\operatorname{vec}(\boldsymbol{\mathcal{T}}) - \boldsymbol{M} \operatorname{vec}(\boldsymbol{\mathcal{I}}_R)\|_2^2 \end{array}. \tag{1.49}$$

But the linear system without constraints is severely illed-posed, and the Kronecker product equation is not trivial to solve either. Considering the constrained linear system is thus a difficult way to write the cost function. Therefore, the cost function to be minimized is chosen to be

$$\gamma(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = \|\boldsymbol{\mathcal{T}} - (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\, \boldsymbol{\Sigma}_R\|_F^2. \tag{1.50}$$

This choice for γ is also problematic because γ is not a convex function. However it is quadratic with respect to each factor. Therefore, a clever way to solve the approximate CPD using only linear systems is to minimize γ over each mode in an alternate fashion, and to use the fixed modes as regressor in a linear system. For example, if the optimization is done on $\boldsymbol{A}$, then $\boldsymbol{B}$ and $\boldsymbol{C}$ are fixed and used as regressors in a linear system $\boldsymbol{T} = \boldsymbol{A}\boldsymbol{X}$ where $\boldsymbol{X}$ depends on $\boldsymbol{B}$ and $\boldsymbol{C}$. When dealing with three way arrays, the three least square problems are obtained by matricizing the array in each mode, since

$$\|\boldsymbol{\mathcal{T}} - (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\, \boldsymbol{\Sigma}_R\|_F^2 = \|\boldsymbol{T}_{(1)} - \boldsymbol{A}(\boldsymbol{B} \odot \boldsymbol{C})^{\mathsf{T}}\|_F^2 = \|\boldsymbol{T}_{(2)} - \boldsymbol{B}(\boldsymbol{A} \odot \boldsymbol{C})^{\mathsf{T}}\|_F^2 = \|\boldsymbol{T}_{(3)} - \boldsymbol{C}(\boldsymbol{A} \odot \boldsymbol{B})^{\mathsf{T}}\|_F^2$$

[1] except for some rare exceptions referenced in [27]

Algorithm 1 using these matricizations is called Alternating Least Squares, and is considered the workhorse algorithm for computing the best approximation of a tensor by a low rank CPD. However, other powerful algorithms have been adapted in the recent years to minimize (1.50), based on gradient descent.

All these algorithms have their advantages and their issues, but we believe [2] that differences in convergence rates and estimation performances mainly lie in the implementation strategy since all methods have similar complexity, and all are believed to converge to one of many critical point of γ depending on initialization. But implementation does matter and precision and computation time may vary a lot depending on how well implementation uses properties of tensor algebra, parallelization, sparsity, the initialization strategy and so on. A nicely implemented toolbox using the Levenberg Macquardt method and Gauss Newton methods is available online[3] while a toolbox for ALS applied to CPD and other decomposition models is due to Rasmus Bro[4]. Another widely used toolbox featuring many basic manipulation functions is the Tensor Toolbox [5]. The authors are using personal MATLAB 2014b codes also available publicly[6].

Algorithm 1 ALS algorithm for solving the exact coupled CPD problem

INPUT: Multiway array $\boldsymbol{\mathcal{T}}$, initials factors $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$.
while convergence criterion is not met **do**

$$\boldsymbol{A} = \boldsymbol{T}_{(1)} \left(\boldsymbol{B} \odot \boldsymbol{C}\right) \left(\boldsymbol{B}^{\mathsf{T}}\boldsymbol{B} \boxdot \boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}\right)^{-1}$$

$$\boldsymbol{B} = \boldsymbol{T}_{(2)} \left(\boldsymbol{A} \odot \boldsymbol{C}\right) \left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} \boxdot \boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}\right)^{-1}$$

$$\boldsymbol{C} = \boldsymbol{T}_{(3)} \left(\boldsymbol{A} \odot \boldsymbol{B}\right) \left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} \boxdot \boldsymbol{B}^{\mathsf{T}}\boldsymbol{B}\right)^{-1}$$

end while
OUTPUT: estimated factors $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$.

To find the best approximate CPD of a multiway array of data is equivalent to finding the structured mean of the distribution (1.47) knowing the noise structure, but with only one realization of this distribution. Clearly since only one realization is available, the final estimate or factors $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ will be biased. However the number of parameters grows linearly with the dimensions of the array whereas the number of equations grows exponentially. Thus with a fairly large array of data following (1.47), a fairly precise estimation of the factors may be obtained in theory. The question of how precise the approximation can be is tackled by computing the Cramér-Rao bound of the approximate CPD under Gaussian i.i.d. noise introduced in Appendix B.

What to do with correlated noise

An issue with ALS methods is that taking correlation into account is seemingly difficult. However in the particular case of array normal distribution, handling correlated noise is straightforward.

Suppose the following probabilistic model for the data array:

$$\boldsymbol{\mathcal{T}} \sim \mathcal{AN}\left(\boldsymbol{\mathcal{X}}, \boldsymbol{\Gamma}\right) \tag{1.51}$$

where $\boldsymbol{\Gamma}$ is an invertible operator in the tensor space of operators, *i.e.* $\boldsymbol{\Gamma} \in \mathbb{R}^{K\times K} \otimes \mathbb{R}^{L\times L} \otimes \mathbb{R}^{M\times M}$. The definition of array normal distribution requires $\boldsymbol{\Gamma}$ to be invertible just like in regular multivariate Gaussian distribution, otherwise a probability density cannot be evaluated with the Lebesgue measure. It also required that $\boldsymbol{\Gamma}$ is a rank one tensor[7], so that there exists $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ symmetric and full rank so that $\boldsymbol{\Gamma} = \boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W}$. This last condition means that the correlation of the noise happens mode-wise, which may be a natural assumption in some contexts. Thus a mode-wise preprocessing of the data is enough to whiten the noise:

$$\left(\boldsymbol{U}^{-\frac{1}{2}} \otimes \boldsymbol{V}^{-\frac{1}{2}} \otimes \boldsymbol{W}^{-\frac{1}{2}}\right) \boldsymbol{\mathcal{T}} \sim \mathcal{AN}\left(\left(\boldsymbol{U}^{-\frac{1}{2}}\boldsymbol{A} \otimes \boldsymbol{V}^{-\frac{1}{2}}\boldsymbol{B} \otimes \boldsymbol{W}^{-\frac{1}{2}}\boldsymbol{C}\right) \boldsymbol{\Sigma}_R, \boldsymbol{\mathcal{I}}\right) \tag{1.52}$$

[2] Since a theoretical comparison of different descent methods for computing and approximate CPD proves very difficult, the only comparisons that exist are comparing simulations of decompositions using these methods, and performances depend on implementation. Thus the fact that all methods are equivalent, and in particular ALS and gradient-based approaches, is simply a belief of ours and is not backed by any evidence other than ignorance.

[3] `http://www.tensorlab.net/`

[4] `http://www.mathworks.com/matlabcentral/fileexchange/1088-the-n-way-toolbox`

[5] `http://www.sandia.gov/~tgkolda/TensorToolbox/index-2.6.html`

[6] `http://www.gipsa-lab.fr/~pierre.comon/TensorPackage/tensorPackage.html`

[7] Be careful that $\boldsymbol{\Gamma}$ belongs to a tensor space of operators, *i.e.* represented by matrices in some canonical basis

and any algorithm designed for i.i.d. noise can be easily extended to cover correlated noise given a mode-wise correlation. This also means that processing a data array with a rank one non-orthogonal operator $\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W}$ correlates the noise mode-wise.

A more difficult situation is encountered when $\boldsymbol{\Gamma}$ is not a rank-1 operator, which is typically the case when data are missing. Then a good definition of an array normal distribution is not even available, but it is still possible to resort to a multivariate normal distribution of the vectorization of $\boldsymbol{\mathcal{T}}$. By using clever matricization properties and reordering the coefficients in $\boldsymbol{\Gamma}$, it is possible to use an ALS algorithm with unstructured correlation [65], but it is also possible to vectorize the data and the CPD model to run a gradient algorithm [2].

There is much more to say about the CPD of a tensor. Many problems encountered both in theory and in practice are developed later in this manuscript, along with other tensor decomposition models. In short,

- including constraints or prior information on factors in the CPD to improve interpretability or ensure existence of the best low rank approximate is the main topic of the next part of this manuscript.
- applications of CPD to real data sets are naively presented in chapter 2, and more complex models based on the CPD are applied in the second part of this manuscript.
- rank estimation is a serious issue in all applications and is shortly evoked along with compression of tensors in section 4.3, but a helpful reference is [20].

1.4.2 Tucker decomposition and High Order Singular Value Decomposition

Tucker decomposition is another widely used tensor decomposition model. Basically, a Tucker decomposition of a tensor $\boldsymbol{\mathcal{T}}$ is a non-invertible change of basis $\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W} \in \mathbb{R}^{K \times R_1} \otimes \mathbb{R}^{L \times R_2} \otimes \mathbb{R}^{M \times R_3}$ applied on $\boldsymbol{\mathcal{T}}$ to obtain a smaller array $\boldsymbol{\mathcal{G}}$ in $\mathbb{R}^{R_1} \otimes \mathbb{R}^{R_2} \otimes \mathbb{R}^{R_3}$ called the core of the decomposition. Similarly to the CPD, an exact Tucker decomposition is not what is typically useful in practice, and an approximation framework is necessary. First the multilinear rank is introduced, which is a key notion to understand the Tucker decomposition. Then a definition of Tucker decomposition is provided along with a discussion on identifiability of the parameters, and the specific case of High Order Singular Value Decomposition is explored.

Multilinear ranks

The rank of a tensor is a notion intertwined with the CPD. For matrices, the tensor rank coincides with mode-wise ranks, *i.e.* column and row ranks. However this is not the case for three way arrays, that is, the tensor rank is not equal to the dimension of the span of rows, columns and fibers (coefficients along the third mode). These three dimensions are called the multilinear ranks of a tensor and are an intrinsic notion to the Tucker decomposition. More discussion on the multilinear ranks can be found in pages 175-180 of [57].

Multilinear ranks of a third order tensor are three integers that express the dimensions of the span of vectors in each mode, *i.e.* the dimensions of the span of rows, columns and fibers. However this definition is not easily understood when dealing with tensors that are not arrays. Thus a coordinate-free definition using the factors of the CPD is provided below.

Definition 14 *Let $\boldsymbol{\mathcal{T}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\, \boldsymbol{\Sigma}_R$ a tensor in $\mathbb{R}^K \otimes \mathbb{R}^L \otimes \mathbb{R}^M$. The multilinear ranks R_1, R_2 and R_3 of $\boldsymbol{\mathcal{T}}$ are the ranks of the factors in its CPD, $R_1 = rank(\boldsymbol{A})$, $R_2 = rank(\boldsymbol{B})$ and $R_3 = rank(\boldsymbol{C})$.*

This is a proper definition since a tensor cannot admit multiple CPDs with factors of different tuples of multilinear ranks. The proof goes as follows. We have to show that given two CPDs of $\boldsymbol{\mathcal{T}}$, the ranks of factors of the same mode are the same. Let $\boldsymbol{\mathcal{T}}$ of rank R and order three admit two different CPDs:

$$\boldsymbol{\mathcal{T}} = \sum_{r=1}^{R} \boldsymbol{a}_r^{(1)} \otimes \boldsymbol{b}_r^{(1)} \otimes \boldsymbol{c}_r^{(1)} = \sum_{r=1}^{R} \boldsymbol{a}_r^{(2)} \otimes \boldsymbol{b}_r^{(2)} \otimes \boldsymbol{c}_r^{(2)}. \tag{1.53}$$

We can consider that all vectors $\boldsymbol{b}_r^{(1)} \otimes \boldsymbol{c}_r^{(1)}$ are linearly independent, otherwise the rank of $\boldsymbol{\mathcal{T}}$ would be smaller than R. Thus we can find a mapping $\phi_{1,1}$ so that $\phi_{1,1}(\boldsymbol{b}_1^{(1)} \otimes \boldsymbol{c}_1^{(1)}) = 1$ and

$\phi_{1,1}(\boldsymbol{b}_r^{(1)} \otimes \boldsymbol{c}_r^{(1)}) = 0$ if $r \neq 1$. $\phi_{1,1}$ is a projection operator on the span of $\boldsymbol{b}_1^{(1)} \otimes \boldsymbol{c}_1^{(1)}$. Then applying $\mathrm{id} \otimes \phi_{1,1}$ on $\boldsymbol{\mathcal{T}}$ proves the consistency of the multilinear rank defintion:

$$\mathrm{id} \otimes \phi_{1,1}(\boldsymbol{\mathcal{T}}) = \boldsymbol{a}_1^{(1)} = \sum_{r=1}^{R} \phi_{1,1}(\boldsymbol{b}_r^{(2)} \otimes \boldsymbol{c}_r^{(2)}) \boldsymbol{a}_r^{(2)} \tag{1.54}$$

which means that $\boldsymbol{a}_1^{(1)}$ is in the span of $\boldsymbol{A}^{(2)}$. A similar reasoning can be done on all vectors in the two CPDs, so that $\mathrm{rank}(\boldsymbol{A}^{(1)}) = \mathrm{rank}(\boldsymbol{A}^{(2)})$, $\mathrm{rank}(\boldsymbol{B}^{(1)}) = \mathrm{rank}(\boldsymbol{B}^{(2)})$ and $\mathrm{rank}(\boldsymbol{C}^{(1)}) = \mathrm{rank}(\boldsymbol{C}^{(2)})$. □

Tensor rank may be larger than the multilinear ranks. An example of third order tensor with rank 3 but multilinear ranks 2, 2 and 2 is introduced below. Given independent vectors $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$,

$$\boldsymbol{\mathcal{T}} = \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 \otimes \boldsymbol{a}_2 + \boldsymbol{a}_1 \otimes \boldsymbol{a}_2 \otimes \boldsymbol{a}_1 + \boldsymbol{a}_2 \otimes \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 \tag{1.55}$$

which is proven to be rank 3 but has factors with column rank 2, since $\boldsymbol{A} = [\boldsymbol{a}_1\ \boldsymbol{a}_1\ \boldsymbol{a}_2]$, $\boldsymbol{B} = [\boldsymbol{a}_1\ \boldsymbol{a}_2\ \boldsymbol{a}_1]$ and $\boldsymbol{C} = [\boldsymbol{a}_2\ \boldsymbol{a}_1\ \boldsymbol{a}_1]$. So it may happen that factors have deficient column rank. In particular, the rank may be larger than the dimensions of an array whereas this is not true for the multilinear rank. However for tensors of small rank R with random factors drawn uniformly $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$, multilinear ranks are equal to the rank with probability 1 since the set of column rank deficient factors has 0 probability. Actually, example (1.55) is a well known particular case of tensors in quantum physics - the maximally entangled tensor of order 3 - and usually does not represent a typical data array.

Exact and approximate Tucker decomposition

Given a tensor $\boldsymbol{\mathcal{T}}$ with multilinear ranks R_1, R_2 and R_3, a natural idea is to project $\boldsymbol{\mathcal{T}}$ onto three subspaces $\mathscr{U}$, $\mathscr{V}$ and $\mathscr{W}$ of dimensions equal to the multilinear ranks. Thus a more concise expression of $\boldsymbol{\mathcal{T}}$ can be achieved. In data mining, these three subspaces are called feature spaces. Changing the basis so that $\boldsymbol{\mathcal{T}}$ is written in the tensor product of three feature spaces $\mathscr{U} \otimes \mathscr{V} \otimes \mathscr{W}$ is exactly what the Tucker decomposition is.

Definition 15 *Let $\boldsymbol{\mathcal{T}}$ in $\mathbb{R}^K \otimes \mathbb{R}^L \otimes \mathbb{R}^M$. Suppose $\boldsymbol{\mathcal{T}}$ has multilinear ranks R_1, R_2 and R_3. Then the Tucker decomposition of $\boldsymbol{\mathcal{T}}$ is given by*

$$\boldsymbol{\mathcal{T}} = (\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\, \boldsymbol{\mathcal{G}} \tag{1.56}$$

where $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ is called the core tensor and $\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W} \in \mathbb{R}^{K \times R_1} \otimes \mathbb{R}^{L \times R_2} \otimes \mathbb{R}^{M \times R_3}$ is a multilinear operator.

It is not possible to write $\boldsymbol{\mathcal{T}}$ without loss in a basis $\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W}$ where the ranks of $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ are smaller than the multilinear ranks R_1, R_2 and R_3. On the other hand, if the multilinear rank is known, it is possible to project on a bigger tensor space than the span of $\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W}$ but this seems pretty useless since this would leave correlated coefficients in the tensor.

However in practice and similarly to the CPD framework, the data $\boldsymbol{\mathcal{T}}$ has large multilinear ranks because it is noisy. In fact since under noise the tensor rank will be generic, the multilinear rank on the first mode of a data array is equal to $\min\left(K, \overline{R}\right)$ where $\overline{R}$ is a generic rank, which in many practical applications equals $\lfloor \frac{KLM}{(K+L+M-2)} \rfloor + 1$. Similar discussion holds for the two other modes. Thus to define an approximate Tucker decomposition, a probabilistic framework may be introduced. Again, three hypotheses have to be verified by the data array $\boldsymbol{\mathcal{T}}$ to successfully apply an approximate Tucker decomposition.

1. There exist an array $\boldsymbol{\mathcal{X}}$ and a noise array $\boldsymbol{\mathcal{E}}$ so that

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{E}}. \tag{1.57}$$

2. Array $\boldsymbol{\mathcal{X}}$ has multilinear ranks R_1, R_2 and R_3 small in comparison to the dimensions of the data, *i.e.* admits and exact Tucker decomposition

$$\boldsymbol{\mathcal{X}} = (\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\, \boldsymbol{\mathcal{G}} \tag{1.58}$$

with $\boldsymbol{\mathcal{G}}$ smaller than $\boldsymbol{\mathcal{X}}$. Since $\boldsymbol{\mathcal{X}}$ stands for the cleaned data, this hypothesis only requires that the data has been generated by a small number of components *mode-wise*, which is a much weaker assumption than asking for a small number of components in the CPD.

3. Array $\boldsymbol{\mathcal{E}}$ has a known probability distribution that does not generate a multilinear structure of small multilinear rank, *i.e.* the multilinear ranks of $\boldsymbol{\mathcal{E}}$ are as high as expected.

In contrast to the CPD, a best low multilinear rank approximation always exists, so that the Tucker decomposition can be used in a context where the existence of $\boldsymbol{\mathcal{X}}$ is not provided by a physical underlying model.

Remark These hypotheses are not equivalent to the CPD hypotheses since a tensor $\boldsymbol{\mathcal{X}}$ can in theory have a high rank but small multilinear ranks. In practice, ranks and multilinear ranks of $\boldsymbol{\mathcal{X}}$ are often equal since its dimensions are larger than its rank. However some models suppose collinear columns in factors of the CPD. In this case the tensor rank and multilinear ranks should be different but the two sets of hypotheses are almost equivalent. In short, stating that a high rank tensor is noisy and that the true noiseless data has low multilinear ranks is enough in practice to apply both Tucker and CPD models given the existence of $\boldsymbol{\mathcal{X}}$.

Identifiability and HOSVD

From a parameter identification perspective, an unfortunate property of the Tucker decomposition is that there are multiple ways to choose a new basis of representation $\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W}$, so that neither these matrices nor the core tensor $\boldsymbol{\mathcal{G}}$ are unique in the decomposition. Indeed,

$$\boldsymbol{\mathcal{T}} = (\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\, \boldsymbol{\mathcal{G}} = (\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\, (\boldsymbol{S}_1 \otimes \boldsymbol{S}_2 \otimes \boldsymbol{S}_3)\, \boldsymbol{\mathcal{G}}' \tag{1.59}$$

for any invertible linear operators $\boldsymbol{S}_i$ so that $\boldsymbol{\mathcal{G}}' = \left(\boldsymbol{S}_1^{-1} \otimes \boldsymbol{S}_2^{-1} \otimes \boldsymbol{S}_3^{-1}\right) \boldsymbol{\mathcal{G}}$. Usually, orthonormal bases are chosen so that

$$(\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\, (\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})^{\mathsf{T}} = \boldsymbol{\mathcal{I}} \tag{1.60}$$

but this does not solve the identifiability problem.

To obtain a unique decomposition, additional constraints imposed on the core array are necessary, so that multilinear transformations resulting in a loss of identifiability cannot be obtained from modifying $\boldsymbol{\mathcal{G}}$. An interesting tool is the High Order Singular Value Decomposition [43]. Bases are chosen orthonormal so as to verify (1.60), but additional constraints on the core are imposed. Slices of $\boldsymbol{\mathcal{G}}$ in all modes are imposed to be orthogonal mode-wise. The main advantage of the HOSVD is that even though the constraints on the core could seem intractable to compute in an optimization algorithm, they are exactly satisfied if computing the Tucker decomposition as in Algorithm 2, which only requires three matrix Singular Value Decompositions (SVD).

Algorithm 2 Algorithm for computing HOSVD with SVDs of the matricizations of $\boldsymbol{\mathcal{T}}$

INPUT: Three-way array $\boldsymbol{\mathcal{T}}$ of sizes $K \times L \times M$, true multilinear ranks R_1, R_2 and R_3

1. Compute truncated SVD of $\boldsymbol{T}_{(1)} = \boldsymbol{U}\boldsymbol{N}_1$ with $\boldsymbol{U} \in \mathbb{R}^{K \times R_1}$
2. Compute truncated SVD of $\boldsymbol{T}_{(2)} = \boldsymbol{V}\boldsymbol{N}_2$ with $\boldsymbol{V} \in \mathbb{R}^{L \times R_2}$
3. Compute truncated SVD of $\boldsymbol{T}_{(3)} = \boldsymbol{W}\boldsymbol{N}_3$ with $\boldsymbol{W} \in \mathbb{R}^{M \times R_3}$
4. Set $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{T}} \bullet_1 \boldsymbol{U}^{\mathsf{T}} \bullet_2 \boldsymbol{V}^{\mathsf{T}} \bullet_3 \boldsymbol{W}^{\mathsf{T}}$

OUTPUT: Core array $\boldsymbol{\mathcal{G}}$, left orthonormal matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$

Even in the presence of noise, Algorithm 2 is known to provide a very good estimation of subspaces of $\boldsymbol{\mathcal{X}}$ although not optimal in the least squares sense [43]. Other constraints on the core can be applied and especially sparsity constraints [23, 91]. However identifiability in this context has not been clearly proved even under some conditions on the spark of the factors.

A final remark is that in the exact Tucker decomposition, all the information about how the modes of the tensor relate to each other is contained in the core $\boldsymbol{\mathcal{G}}$, meaning that the matrices $\boldsymbol{U}, \boldsymbol{V}$ and $\boldsymbol{W}$ only contain mode-wise information. On the contrary, the factor matrices in the CPD decomposition mine solely "multilinear" information, that is, the CPD answers question like "what are the elementary links between mode on and mode two under conditions set by mode three".

That is why the Tucker decomposition is sometimes said to be a nice tool for processing arrays of data, *e.g.* for compression, but not a meaningful multilinear data mining model like the CPD. Also, the Tucker model does not allow for identifying parameters.

Chapter 2

Applications of tensor decompositions

Summary In this chapter, some applications of tensor decomposition models are described. In particular, the growing importance of tensor models in chemometrics and hyperspectral image processing is supported through the use of physics-based models, and the advantages and challenges of applying multiway data mining models in these areas are explored. Moreover, the application of tensor decomposition techniques to Electroencephalography (EEG) data is quickly surveyed. This chapter is structured as follows:

- Explanations are provided on why fluorescence spectroscopy is a common application of the low rank CPD. Some open challenges and recent works on non-linearity are also surveyed.
- We link tensor decomposition methods and the spectral unmixing problem. Hyperspectral images have many features that call for specific decomposition models, raising some open questions.
- Applications of tensor decomposition models to source extraction from EEG data are overviewed.

This chapter is based on joint work with Xavier Luciani [29, 34] for the non-linear tensor decomposition applied to fluorescence spectroscopy, and with Miguel Veganzones for the use of tensors in spectral unmixing [116–118]. Moreover, the neuroscience section is motivated by a recent work with Bertrand Rivet [99] and ongoing collaboration with Morten Mørup.

2.1 Chemometrics

Chemometrics is the application of data mining tools to data collected in chemistry related experiments. It is of course an immensely wide and highly cross-disciplinary topic. In what follows, only one measurement type is studied, namely fluorescence spectroscopy. As explained below, fluorescence data fits rather well the approximate CPD model described in section 1.4.1. Pioneer works in applying tensor decomposition models to fluorescence spectroscopy are due to Andersson, Bro, Kiers and Smilde [15].

2.1.1 Fluorescence spectroscopy experimental setting

In fluorescence spectroscopy, the object of study is a mixture of several fluorescent chemical components. Each of these chemical components have a characteristic spectral response to light stimulation. This means that for an excitation of a component by a laser beam of wavelength λ_{ex}, the component emits a characteristic spectrum $s_{em}(\lambda_{em})$ called the emission spectrum, where λ_{em} belongs to a certain range. Ideally, only the amplitude of the emission spectrum depends linearly and continuously on λ_{ex}. The concatenation of all these amplitude values makes up for the excitation spectra $s_{ex}(\lambda_{ex})$, where λ_{ex} also belongs to a narrow range outside which the amplitudes are negligible. Figure 2.1 gives an example of emission and excitation spectra estimated through tensor decomposition.

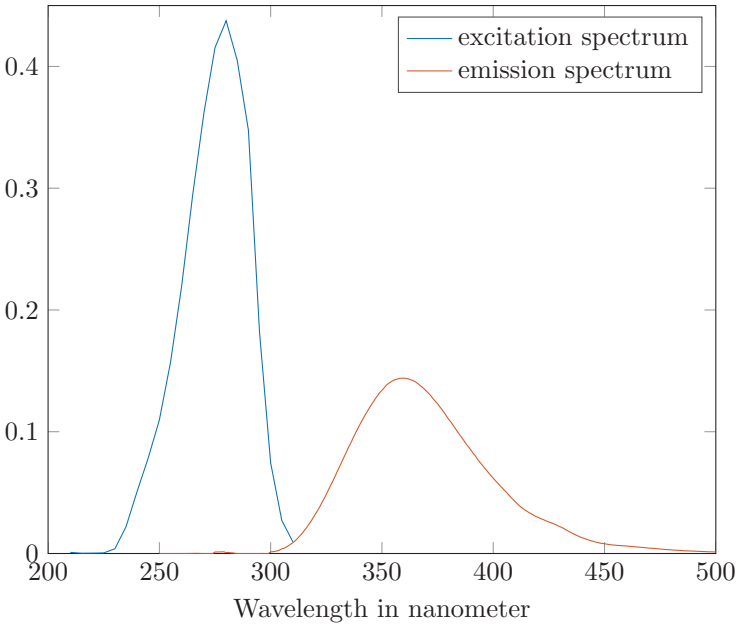


Figure 2.1: Emission and excitation spectra of a peptide, Tryptophan-Glycine, estimated from the data set introduced in section 2.1.2. The blue and red curves correspond respectively to the first column of factors $\boldsymbol{A}$ and $\boldsymbol{B}$ in model 2.2. Note that these estimated spectra may have various shapes depending on some unreported parameters, notably concentration, see 2.1.3.

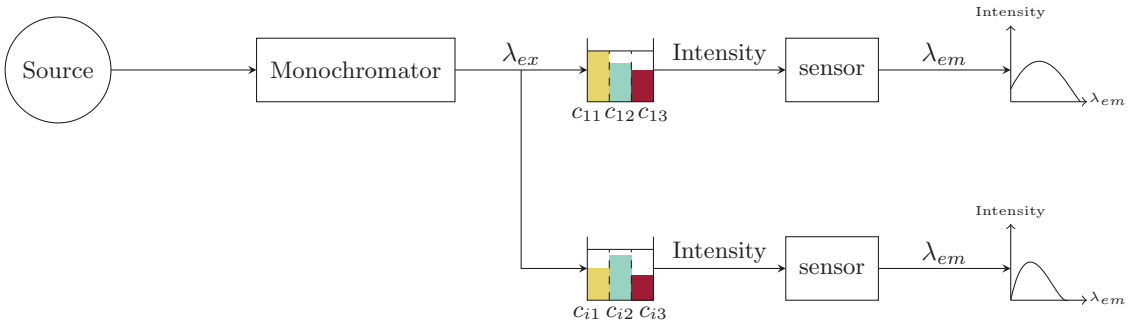


Figure 2.2: A typical experimental setting for fluorescence spectroscopy measurements with multiple experiments along the concentration diversity.

Another characteristic of fluorescence spectroscopy is that, should multiple components be present in a studied mixture, the emission spectrum emitted by the mixture is the sum of each individual emission spectrum of each component with independent amplitudes. This means that we know how the spectra are mixed and scaled when a mixture of chemicals is hit by a laser beam with known wavelength. Figure 2.2 shows a typical experimental setting for fluorescence spectroscopy. In practice, the data are made of values of intensity, that stand for quantified values of the mixed emission spectra on a grid of wavelengths. Note that not all chemical components are fluorescent, so that when studying mixtures of chemicals with fluorescence spectroscopy, some components may not contribute to the final emission spectrum.

In terms of data, one mixture provides with one matrix of intensities called the Fluorescence Emission Excitation Matrix (FEEM). If there is no noise corruption and if the previous hypothesis are exactly verified, then a FEEM $\boldsymbol{M}$ is a low rank matrix:

$$\boldsymbol{M} = \boldsymbol{A}\boldsymbol{\Sigma}_R\boldsymbol{B}^\mathsf{T} = \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r \otimes \boldsymbol{b}_r \tag{2.1}$$

where columns of factor matrix $\boldsymbol{A}$ are the emission spectra for each component in the mixture, columns of factor matrix $\boldsymbol{B}$ are the excitation spectra for each component, R is the number of

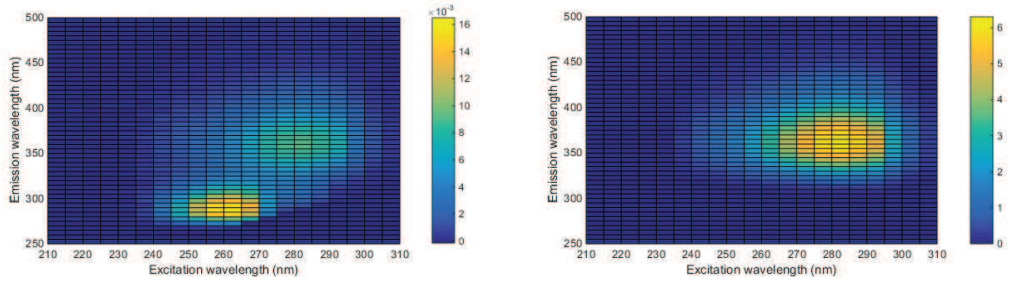


Figure 2.3: Left : FEEM obtained from the 8th experiment from the data set described in section 2.1.2. Right : A normalized reconstructed FEEM matrix for the Tryptophan-Glycine. Warm colors stand for large intensity of emitted light and cold colors for low or no intensity.

components in the mixture, and λ_r is some scaling depending on multiple parameters. As explained in section 1.4.1, there is not a unique low rank decomposition for second-order tensors, and this means that from only one FEEM the actual emission and excitation spectra cannot be recovered solely using low rank methods. To overcome this difficulty, a solution is to add a concentration diversity in the measurements, *i.e.* measure the response to laser stimulation of various mixtures containing the same components at different concentrations[1]. At low values, concentration acts linearly on the emission and excitation spectra, which means that the measured intensity will be twice higher for a component which is twice more concentrated in the mixture. As explained below, this hypothesis is true only if the absorption of the solution can be neglected, which is often the case at low concentrations. Then due to the linearity on the concentration mode, the multiple FEEM can be stacked into a multiway array $\boldsymbol{\mathcal{T}}$ that follows a low rank model:

$$\boldsymbol{\mathcal{T}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\, \boldsymbol{\Sigma}_R = \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r \otimes \boldsymbol{b}_r \otimes \boldsymbol{c}_r \tag{2.2}$$

where columns of factor $\boldsymbol{C}$ are the relative concentrations of one component along each experiment (*i.e.*, the relative height of the columns of same color in Figure 2.2). In practice, observation noise and modeling errors have to be accounted for, so that additive noise is added to model (2.2).

In short, if an approximate CPD is applied to fluorescence data, the rank stands for the number of fluorescent components in the mixture, and the extracted sources are respectively the excitation spectra, the emission spectra and the concentration profiles of each component in each mixture. Because these factors have physical meaning, some information is a priori available. First, all components are non-negative, meaning that the extracted components should not have negative coefficients. Moreover, the emission and excitation spectra are likely smooth in the range of measurements.

2.1.2 Data for simulations

Throughout this manuscript, one data set stemming from chemometrics will be used. This data set was developed by Acar et. al. in the context of data fusion [4]. For this data set, five chemical components are mixed at different concentrations: Valine-Tyrosine-Valine (Val), Tryptophan-Glycine (Gly), Phenylalanine (Phe), Maltoheptaose (Mal) and Propanol (Pro). The original data are composed of three measurement modalities, namely fluorescence spectroscopy (EEM), nuclear magnetic resonance (NMR) and liquid chromatography - mass spectroscopy (LC-MS), but only the fluorescence and NMR data are used here. The concentration profiles for the five chemicals are shared by all experiments and are given in table 2.1. 28 mixtures are studied.

For the EEM data, excitation wavelengths vary from 210nm to 310nm with intervals of 5nm while emission wavelengths are recorded from 250nm to 500nm with increments of 1nm. This means that the EEM tensor has sizes $21 \times 251 \times 28$. Only three chemicals used in the experiment are fluorescent, so that Maltoheptaose and Propanol cannot be observed in the EEM data. This means that the noiseless rank of the EEM tensor is 3. Reference spectra for Val, Gly and Phe are not provided in the data set, but some estimated spectra have been provided by the authors and are presented further in Figure 4.4 of section 4.2.6.

[1] In chemometrics this is called second order calibration

Mixture	Val	Gly	Phe	Mal	Pro
1	5.00	0	0	0	0
2	0	5.00	0	0	0
3	0	0	5.00	0	0
4	0	0	0	5.00	0
5	0	0	0	0	5.00
6	1.25	5.00	3.75	3.75	0
7	3.75	1.25	5.00	1.25	0
8	2.50	5.00	2.50	1.25	0
9	5.00	3.75	2.50	3.75	0
10	3.75	3.75	5.00	0	1.25
11	6.25	1.25	1.25	0	2.50
12	1.25	.005	2.50	0	5.00
13	2.50	6.25	2.50	0	2.50
14	5.00	1.25	3.75	1.25	3.75
15	1.25	2.50		2.50	2.50
16	3.75	1.25	5.00		1.25
17	2.50	0	1.25	1.25	6.25
18	1.25	0	5.00	2.50	0
19	3.75	0	2.50	5.00	0
20	2.50	0	3.75	1.25	0
21	5.00	0	1.25	3.75	0
22	3.75	0	3.75	0	5.00
23	5.00	0	1.25	0	3.75
24	1.25	0	5.00	0	2.50
25	2.50	0	2.50	0	1.25
26	5.00	0	1.25	3.75	3.75
27	3.75	0	5.00	2.50	1.25
28	2.50	0	2.50	1.25	5.00

Table 2.1: Concentration in milliMolar, from [4].

The underlying process that generates the NMR data is not reported in this manuscript, further information on the NMR system itself can be found for instance here [53]. In short, the idea is similar to fluorescence spectroscopy, but instead of fluorescence, the physical process that generates the recorded data is nuclear magnetic resonance. The three diversities are the chemical shift, the gradient levels and the relative concentrations of the chemical components in the mixtures. The chemical shifts are expressed in part per million (ppm) and informs on the structure of the molecule. The location of the peaks of chemical shifts is characteristic of the chemical environment of the protons. Moreover, according to [4], "the gradient levels encode the diffusion property of the various molecular species". There are 8 recorded gradient levels, and 13324 chemical shifts. The NMR tensor data are therefore $13324 \times 8 \times 28$. The fact that NMR data follows a low rank approximate model is only checked here through the good results of experiments conducted in further chapters.

Notably, the fluorescence spectroscopy modality features many missing data. Missing data may have multiple origins, but in fluorescence spectroscopy, they are often caused by Rayleigh and Raman artifact removal [98]. Missing data is not a topic explored in this manuscript, although some works on imputing the missing values exist in the litterature [2, 16]. Here, missing data were simply replaced by realizations of the absolute value of a random variable following a Gaussian distribution of zero mean and variance 10^{-4}.

2.1.3 A non-linear decomposition for fluorescence data

When using low rank techniques to extract information on the spectra and concentration from fluorescence data, the usual hypothesis is that the linear model fits the data well, and that any modeling error is minor. The reasoning behind this hypothesis is that physical equations linking measured intensity, spectra and concentrations can be linearized at low concentrations. The linearization process is clearly explained by Luciani *et. al.* in [85]. However, it is shown in the same paper that at higher concentrations, linearization comes at a loss of precision, *i.e.* the multilinear

model is not appropriate anymore. A first solution is to dilute the mixtures in order to have low concentrations at all time when using fluorescence spectroscopy, but dilution may deteriorate the mixture.

A second solution is to fit the non-linearized model described by the following equation:

$$T_{klm} = \sum_{r=1}^{R} a_{kr} b_{lr} c_{mr} \prod_{p=1}^{R} e^{-c_{mp}\left(b_{lp}+b'_{kp}\right)} \tag{2.3}$$

where $\boldsymbol{B}'$ is zero-padded to account for coefficients b_{kr} where $k > L$. Note that this model is a coefficient-wise multiplication of a usual CPD with an exponential correction term, depending on the concentration and the excitation spectra. This exponential attenuation factor emerges from the Beer-Lambert modeling of the absorbance of the solution, also called inner-filter effects. When concentrations are low, clearly the correction term tends to 1, so that the data tensor almost follows a CP decomposition model. We call this model the Non-Linear Fluorescence Decomposition (NLFD).

NLFD is thoroughly studied in [34], only the main results are summarized here. First the scaling ambiguity among the parameters of the NLFD is different from usual CPD. Indeed, only factors $\boldsymbol{C}$ and $\boldsymbol{B}$ have free column norms, but columns of factor $\boldsymbol{A}$ cannot be scaled at wish:

$$\begin{aligned} T_{klm} &= \left(\sum_{r=1}^{R} a_{kr} b_{lr} c_{mr} \right) \prod_{p=1}^{R} e^{-c_{mp}(b_{lp}+b'_{kp})} \\ &= \left(\sum_{r=1}^{R} a_{kr} \tfrac{b_{lr}}{\lambda_r} \lambda_r c_{mr} \right) \prod_{p=1}^{R} e^{-\lambda_p c_{mp} \frac{(b_{lp}+b'_{kp})}{\lambda_p}} \end{aligned} . \tag{2.4}$$

Moreover, it is checked numerically in [29] that the NLFD is not identifiable when only two modalities are provided. Indeed, it is shown in Proposition 2 of [29] that for matrices, non-linearity does not restore the identifiability of the parameters in the matrix decomposition model.

To actually compute the NLFD, since an ALS algorithm seems difficult to design, a Levenberg-Marquardt descent algorithm inspired from [86] is used instead. Moreover, the NLFD written in 2.3 is not exact if the excitation wavelengths and the emission wavelengths do not perfectly match. Indeed, $\boldsymbol{B}'$ is a function of λ_{em} but is sampled on the grid of λ_{ex}, and the two grids may not coincide so that the values of $\boldsymbol{B}'$ are not contained in $\boldsymbol{B}$. This problem is tackled efficiently by any interpolation method, say linear interpolation, but the NLFD has to be slightly modified to account for this interpolation.

It is shown both on simulations and on actual fluorescence spectroscopy data in [34] that NLFD outperforms regular CPD when concentrations are not low. A lesson that can be drawn from these results is that trying to fit a CPD to any tensor data without understanding its true structure is typically a terrible idea. Only when the approximate low rank model fits the data well, or is backed up by a physical model, should the CPD be computed. Otherwise any interpretation of the estimated factors is very subjective. In that case, studying the physical properties of the process generating the data may provide a modeling solution, as was introduced in this section.

2.2 Hyperspectral image processing

Hyperspectral and multispectral imaging is concerned with the acquisition and processing of images recorded at multiple wavelengths. For instance, a usual RGB image is multispectral in the sense that it contains light intensity data in two (spatial) way arrays for three wavelengths. In general, hyperspectral images are 2 dimensional scenes recorded for hundreds of wavelengths (see Figure 2.4), while multispectral images typically have a dozen of wavelengths. Throughout this manuscript, we do not distinguish multispectral and hyperspectral since this has little importance to what is discussed below. Thus only the terminology "hyperspectral imaging" is used. In what follows, light is shed onto why approximate CPD can be applied to hyperspectral imaging only when a fourth way is available, *e.g.* time, angle or patches. Some challenges emerging from the spectral unmixing application are surveyed, and the "Snow" data set is described for later use. More information on general hyperspectral imaging can be found here [8].

Hyperspectral images are snapshots of a given scene at multiple wavelengths, so that the raw data are usually a three-way array where two dimensions are the spatial dimensions and the third way stands for the various recorded wavelengths. What is measured is the amount of sunlight that

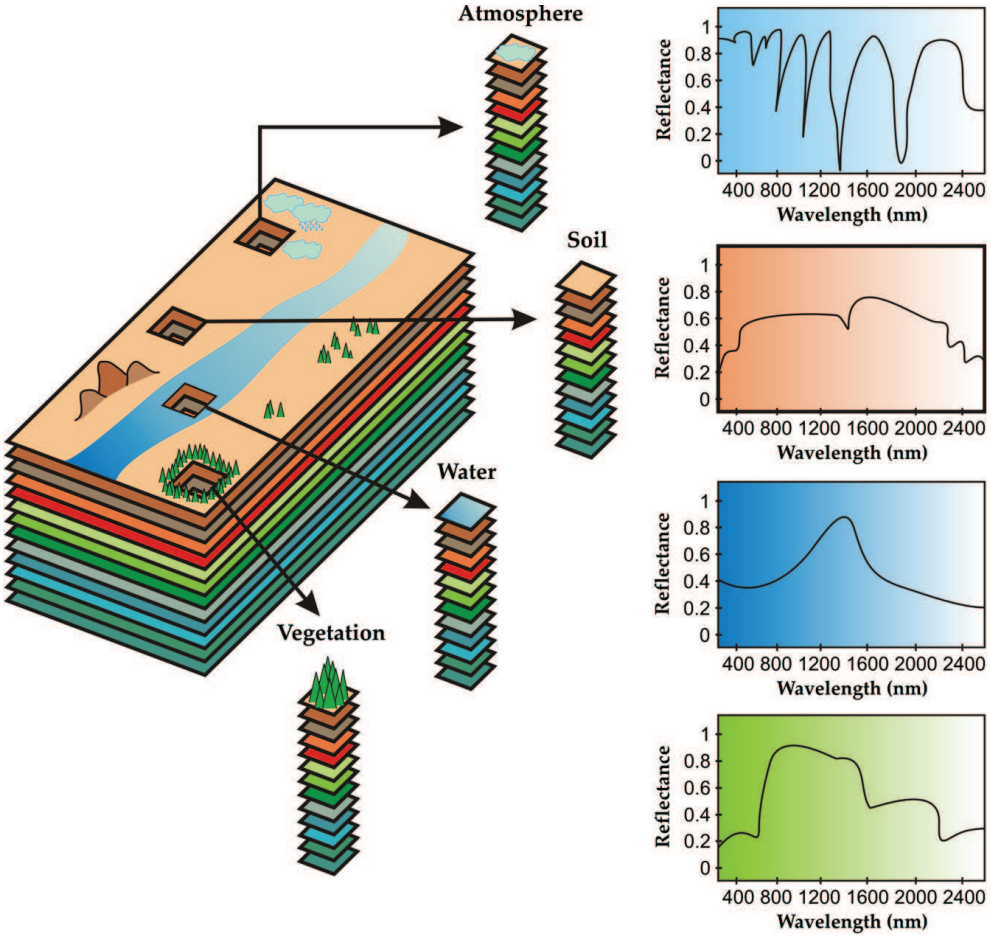


Figure 2.4: An example of hyperspectral image, from [8].

is reflected by each material in the scene called the luminance, which is then processed to obtain reflectance data. The technology for acquiring these images has drastically improved in the recent years, which explains the recent surge of research papers in this area and makes it an appealing research topic for data scientists.

2.2.1 On the multilinear modeling of images

Before giving a tensor decomposition model for hyperspectral data, it is first of utmost importance to stress out when hyperspectral data can be considered as a high-order tensor approximately of low rank. Indeed, a naive way to build a tensor from an hyperspectral image would be to consider that the two spatial ways make up for two modalities, and the spectral way stands for the third diversity. Then the three-way array defining the hyperspectral image is decomposed using approximate CPD, and extracted components on the wavelength mode are understood as reflectance spectra. A similar approach based on Tucker decomposition has even been used in recent research papers [92].

But there is at least two major loopholes in this modeling approach for hyperspectral images. First, note that in a low rank CPD, each sub-tensor obtained by fixing an index on one mode is a low rank tensor of reduced order. Actually the rank of the sliced obtained is necessarily smaller than the rank of the whole tensor. For instance, if third-order $\boldsymbol{\mathcal{T}}$ follows a CPD with rank R, then for $T^m_{kl} := T_{klm}$,

$$\boldsymbol{T}^m = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{c}_m)\, \boldsymbol{\Sigma}_R = (\boldsymbol{A} \otimes \boldsymbol{B})\, \boldsymbol{\Sigma}'_R \tag{2.5}$$

where $\boldsymbol{c}_m$ is the m-th row of $\boldsymbol{C}$ and $\boldsymbol{\Sigma}'_R = \mathrm{diag}\{\boldsymbol{c}_m\}\boldsymbol{\Sigma}_R$. As a consequence, to apply a CPD or Tucker decomposition technique to an hyperspectral image amounts to supposing that each image of the data set is low rank. While it is certainly possible that some images are low rank inside a set of pictures, typically images are not low rank. This can be checked easily by any interested reader

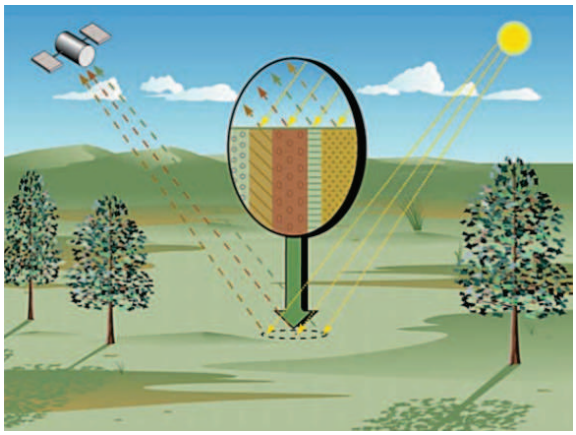


Figure 2.5: A context where the LMM applies. Note that the light reflects on each material, and that the mixing is only effective at the sensor. The illustration is a courtesy of Miguel A. Veganzones.

by computing the singular values of any image that he believes should be low rank. Since images are not low rank, the two spatial dimensions do not offer two meaningful diversities. Instead, an hyperspectral image understood as a third-order tensor is typically of high rank, and applying low rank techniques like Tucker decomposition or CPD means seriously ignoring the data structure. To be fair, it is possible that the multilinear rank is strictly smaller than the tensor rank, so that applying multilinear tools to hyperspectral images could prove useful in theory and under particular care. But for discrepancy between tensor rank and multilinear rank to occur in natural images is doubtful. That is why hyperspectral images are considered as second order tensors in this manuscript, the spatial way being the vectorization of the two original spatial dimensions. This also means that to use tensor decomposition tools in hyperspectral image processing, a third diversity is required. Some examples are provided below in section 2.2.2.

Another issue with hyperspectral images is whether or not they even follow an approximate second-order low rank model after the spatial ways are flattened. The questions asked here are how well does a low rank CPD fit hyperspectral images, and can recovered factors in a CPD be interpreted physically ? Of course these two questions arise for any application of low rank decomposition techniques, and it is often difficult to assess the interpretability of the results. In hyperspectral image processing, a low rank model should however provide with a good tool for what is called spectral unmixing, *i.e.* for estimating the concentrations (called abundances) and spectra of the different materials present in the scene. Indeed a first order physical model [58] links the abundances and the spectra to the measured reflectance in a bilinear fashion. For an hyperspectral image with N pixels and L wavelengths,

$$\boldsymbol{M} = \boldsymbol{A}\boldsymbol{I}_R\boldsymbol{B}^\mathsf{T} = \sum_{r=1}^{R} \boldsymbol{a}_r \otimes \boldsymbol{b}_r \tag{2.6}$$

where $\boldsymbol{a}_r \in \mathbb{R}_+^N$ stands for the abundances of material r in each pixels, $\boldsymbol{b}_r \in \mathbb{R}_+^L$ is the reflectance spectrum of material r, and R is the total number of material. This model relies on the hypothesis that the sunlight reflects on each material independently (see Figure 2.5), and implies that the reflectance of each material at all wavelengths is linearly dependent to the proportion of this material on each pixel. This model is called the Linear Mixing Model (LMM) in the literature, although equation (2.6) is a bilinear model. LMM is typically unrealistic when the mixture of materials in the scene is intimate [10], but is nonetheless quite accurate for wide pictures like images collected from a satellite or an airplane.

As explained in section 1.4.1, it is not possible to identify factors $\boldsymbol{A}$ and $\boldsymbol{B}$ in (2.6) without further assumptions. The non-negativity of factors is often used to restore identifiability [55]. It is also possible to use the fact that rows of $\boldsymbol{A}$ should sum to one, since the sum of relative concentrations should be one [10]. But if additional data are provided along the images, it can be used to add a diversity to the hyperspectral data and use tensor decomposition techniques. Below are introduced some possible additional diversities.

2.2.2 Examples of third order hyperspectral data

Time diversity

To add a third diversity to hyperspectral data, a first solution is to capture hyperspectral images at multiple instants [116]. The raw data are then a four-way array but is seen as a third order tensor $\boldsymbol{\mathcal{T}}$ in $\mathbb{R}_+^N \otimes \mathbb{R}_+^L \otimes \mathbb{R}_+^T$ where T is the number of hyperspectral images collected. An example of such space/wavelength/time data is introduced below in section 2.2.4. Since a collection of images along time is usually called a movie, $\boldsymbol{\mathcal{T}}$ can be designated as an hyperspectral movie.

The main question to be asked here is whether $\boldsymbol{\mathcal{T}}$ follows an approximate CPD with small variance or not, or under which conditions. Unlike the LMM, it is not known if a physical model can justify the low rank CPD. However, a physical interpretation of a low rank CPD applied to $\boldsymbol{\mathcal{T}}$ can be given, which does not prove the approximate CPD model will fit the data well but at least explains what is possibly obtained as the time factor $\boldsymbol{C}$ after computing the decomposition. Suppose the approximate CPD fits the data appropriately, *i.e.*

$$\boldsymbol{\mathcal{T}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\, \boldsymbol{I}_R + \boldsymbol{\mathcal{E}}. \tag{2.7}$$

Then column r of $\boldsymbol{C}$ can be interpreted as the evolution of the rth material distributed on all images according to the abundances contained in the rth column of $\boldsymbol{A}$. More precisely, the time evolution has to be independent of the spatial distribution of the material. This means that movement is not accounted for in the model, and that the material may not partially disappear at multiple pixel locations and at multiple time frames.

Since these hypothesis are extremely strong, most hyperspectral movies will not follow an approximate CPD of rank R where R is exactly the number of different materials in the scenes. Instead, by increasing the rank of the decomposition, the model is relaxed to let multiple components stand for the same material but with different time evolution and abundances. This justifies the use of particular decomposition methods for mining hyperspectral movies, where colinearity between columns of the spectral factor $\boldsymbol{B}$ is taken into account. Furthermore, the abundances have to change significantly along the time axis, so that time really stands for a third diversity. This means that the sampling rate over time must be consistent with the rate at which the landscape is changing, *e.g.* one image a day for snapshots of the Alps. Moreover, estimating the rank of the decomposition proves quite challenging since it is not directly related to the number of materials in the scenes anymore.

Angle diversity

Similarly to adding a time diversity, it is possible to add an angle diversity. That is, multiple images of the same scene are captured but at different capture angles. The measured reflectance of materials is highly dependent on the surface topography, for instance a flat field of grass will give a brighter signal than grass on a mountain area with large slope variations. The angle modality captures these variations and therefore relates to the topography of the surface. It is also likely that some components will have the same spectral factor but with different angular factors and abundances that stand for the various sloppy areas of the scene. The application of approximate CPD to this kind of data is discussed in [117].

Patches diversity

To account for local variability in images, patches have been introduced successfully in image processing. Patches are built by extracting small sub-images from the original image. They can be used in a wide variety of ways, including dictionary learning [87], clustering or denoising [48]. As explained above, images are typically high rank, but flattening along the spatial way is also a suboptimal way to deal with the spatial diversity since some information is contained in the two-way structure of images. Patches can be used to build a third diversity which takes into account two-dimensional variations in the image, while preserving the low rank structure of the flattened hyperspectral image [118]. Indeed it is shown below how to relate the patch-array's approximate CPD with a regularized extended LMM.

First, let us define the patch-array built from an hyperspectral image. Denote by $\boldsymbol{x}_i \in \mathbb{R}_+^L$ the i-th pixel of an hyperspectral image. For each pixel $\boldsymbol{x}_i$, a patch is defined by the matrix $\boldsymbol{P}_i = [\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{x}_{i_B}]$, of B pixels $\boldsymbol{x}_{i_k}$ in the neighbourhood of $\boldsymbol{x}_i$, including $\boldsymbol{x}_i = \boldsymbol{x}_{i_1}$. A

neighbourhood of $\boldsymbol{x}_i$ can be the set of pixels adjacent to $\boldsymbol{x}_i$, see Figure 2.6. Each matrix $\boldsymbol{P}_i \in \mathbb{R}_+^{L \times B}$ is stacked rowwise to define a slice of the patch-array, $\boldsymbol{\mathcal{X}} \in \mathbb{R}_+^{N \times L \times B}$. The above formulation of the patch-tensor is equivalent to stacking B images $\boldsymbol{X}^{(k)} \in \mathbb{R}_+^{N \times L}$ along the third mode, $1 \leqslant k \leqslant B$. Images $\boldsymbol{X}^{(k)}$ are obtained by shifting the original hyperspectral image along the neighbourhood, *i.e.* a shifted image is defined by $\boldsymbol{X}^{(k)} = \left[\boldsymbol{x}_{(1+d_k)}, \ldots, \boldsymbol{x}_{(N+d_k)}\right]^\top$, where d_k denotes a spatial displacement from the center of the patch to the position of the k-th element in the patch. This yields zero-padding along the border of the hyperspectral image.

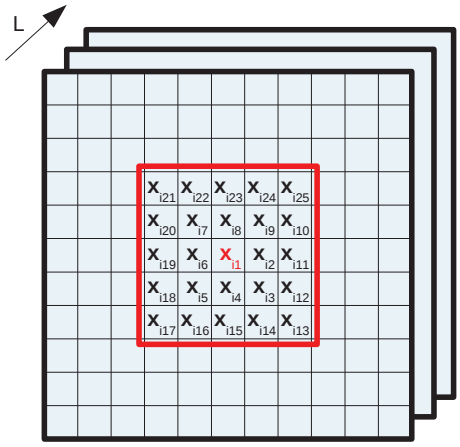


Figure 2.6: Example of a 5×5 patch using a sliding window centred in $\mathbf{x}_i$. The arrow indicates the spectral way.

It is possible to derive the approximate CPD of the patch array from a regularized LMM. LMM does not take into account any two-dimensional structure of the image, and a way to account for this structure is to add a proximity term relating the abundances and spectra of close pixels. Moreover, LMM also fails at modeling spectral variability, which is the variation in amplitude of the spectra along pixels. Spectral variability is also supposedly independent of the abundances. Since spectral variability is mostly a local phenomenon that lives in the two-dimensional structure of the image, it has been shown that a good way to model this variability is to add a scalar parameter to the spectra [52], which can further be defined only on the patches, giving birth to the following regularized extended linear mixing model:

$$\begin{cases} \boldsymbol{x}_i = \sum_{r=1}^{R} a_{ir}\boldsymbol{b}_r + \boldsymbol{e}_i \\ \boldsymbol{x}_{i_j} = \sum_{r=1}^{R} \lambda_{jr} a_{ir}\boldsymbol{b}_r + \boldsymbol{e}_{i_j} \end{cases}. \tag{2.8}$$

The regularization term is induced by the second equation relating a pixel with the spectral unmixing of its neighbors. The variances of errors $\boldsymbol{e}_i$ and $\boldsymbol{e}_{i_j}$ can be modified to tune the amount of regularization imposed on the extended LMM. If all variances are equal, then the approximate CPD of the patch-array with that variance is exactly the maximum likelihood estimation of all parameters in model (2.8) where the first row of the factor related to patches is set to 1.

2.2.3 Challenges

The difficulties inherent to hyperspectral image processing using tensor decomposition are quite different from the issues encountered in chemometrics.

Interpretability

First, interpreting the results is usually more difficult. Indeed, the multilinear model for hyperspectral images with a third provided diversity is not backed by a physical model (at least in the cases explored above). This means that there is no reason to expect that the estimated factors stand for respectively abundances, spectra and a third diversity. Constraining the set of admissible solutions is a possible way to ensure that the extracted factors satisfy some important properties

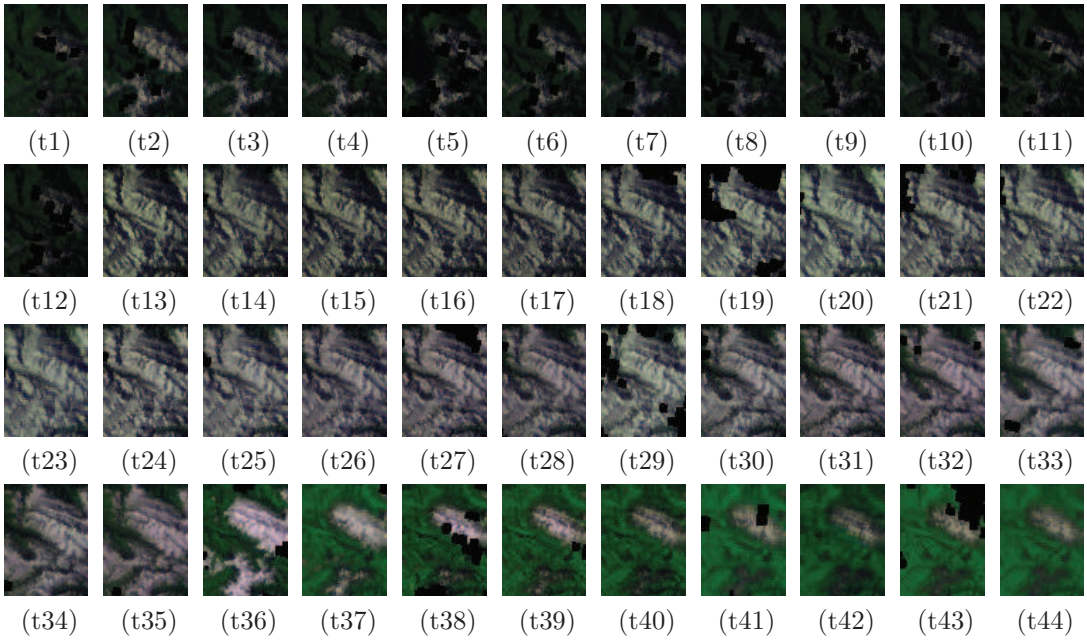


Figure 2.7: False color images of the 44 time acquisitions (cloud pixels are depicted in black). Note the apparition and melting of snow along time, with at least one permanent snowy zone at the summit and intermittent grass-snow zones around it. Clearly the evolution of abundances of snow over time cannot be represented by a single component but at least by two.

of the physical underlying sources. This point is discussed in the dictionary-based CPD in section 3.3.

Big Data

Another common issue with hyperspectral image processing is the dimensions of the tensor decomposition problem. Time, angle or patches often induce a third way of small dimension, but the wavelength way is usually quite large (10^2 to 10^3) while the pixel dimension can be dramatically huge (more than 10^6). Setting aside the storage of the data, the computation of the CPD of large constrained tensors is computationally costly. Some solutions are studied in section 3.2.1 that resort to compression, which however suppose that the HOSVD of the data can be somehow computed.

2.2.4 Snow data

Along with the fluorescence data set described above, a set of hyperspectral images is used in this manuscript. This data set labeled as the "Snow data" is a subset of a longitudinal daily acquisition of MODIS hyperspectral sensor (seven spectral bands) for the same scene in the Alps (France) during the 2012 snow season. The data has been pre-processed to improve the spatial resolution to 250m. From the original data set 44 acquisitions are selected with a cloud presence lower than 30% [2]. Remaining missing data are replaced by absolute values of a white Gaussian random variable of negligible power. Each image is of 80×60 pixels size with seven spectral bands measuring the radiance at the sensor.

The Snow data is quite small since it is a subset of a bigger acquisition. It is deliberately chosen small so that classical decomposition methods can be used to compute the approximate CPD, while it is still large enough to illustrate the advantages of using fast compressed methods introduced in section 3.2.1. Moreover, a dictionary of eight spectra expected to be found in the observed scene is provided by French weather monitoring institute Meteo France. Yet the ground-truth abundances and spectra are not known.

[2] Clouds completely hinder the spectral measurements and therefore induce missing data.

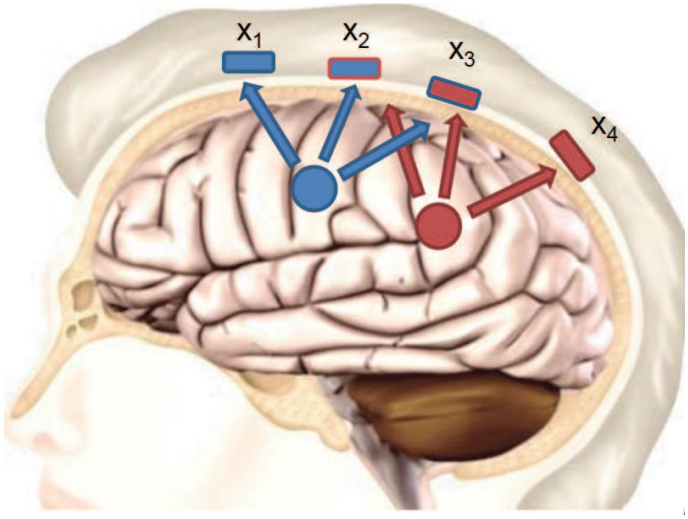


Figure 2.8: An experimental setting for measuring EEG signals. The electrodes are located all over the scalp to ensure that brain activity is spatially equally measured. The illustration is a courtesy of Louis Korczowski.

2.3 Electroencephalography data

Electroencephalography (EEG) is a measurement device used to measure brain electrical activity that was first used in the 1920's. A binary number of electrodes are placed on the head (scalp) of an individual at predetermined localizations (see Figure 2.8). These electrodes collect electrical intensity over time at a high temporal resolution. Because neural activity induces characteristic electrical responses depending on the task accomplished by the subject, it is possible in theory to identify these characteristic responses in the temporal activity measured by EEG, thus collecting precious information on the processes taking place inside the brain of the subject. An example of characteristic response is the P300, which is generated when the subject is aware of doing a task. The shape of the P300 may vary between subjects, but it is globally known and can be observed in the data, as proven by recent experimental designs [40, 41, 76].

EEG measurements for one subject consist of two-way data. Electrical intensity is measured through time at multiple spatial locations. So the collected data are a two-way array $\boldsymbol{M} \in \mathbb{R}^{N \times M}$ where the N rows consist of measured temporal responses at M times slots for each electrode. It is believed that a linear mixing model for the sources and their spatial distribution on the scalp is an adequate model [40], and a multitude of linear source separation methods have been designed to extract meaningful factors so that

$$\boldsymbol{M} = \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{B}^{\mathsf{T}} \tag{2.9}$$

where columns of $\boldsymbol{B}$ are the characteristic time responses, and columns of $\boldsymbol{A}$ represent their spatial distribution on the scalp. The scaling $\boldsymbol{\Sigma}$ refers to the relative power of these sources. While the temporal extracted sources can be interpreted directly, the spatial distribution of the sources is only obtained on the scalp, so that some additional post-processing (*e.g.* sLORETA [94]) is necessary if one seeks further information on the localization of the sources inside the brain.

Because only two modalities are available from EEG data, it is again necessary to use a third diversity to apply tensor decomposition techniques. Many approaches have been developed in the literature to obtain a third diversity, most of which are surveyed in [5], but let us explore two in particular. First, multiple trials can be stacked to form a cube of data where each frontal slice contains the two-way data of one trial. The trial diversity can refer to different subjects or experimental setting [73], or truncated sequences of interest of one measurement [99]. In this context a simple CPD is however difficult to justify because of the variations that exists between the same sources emitted at different trials [89, 90]. Exploring how to tackle this variability directly in the decomposition models is the subject of chapter 4. Another way to obtain a third modality is to extract features from the signal. For instance, if the temporal recorded signals are non-stationary, then a time-frequency study of the EEG data should reveal some information. In this spirit, the windowed wavelet transform of the temporal signals was used in [7]. In both cases, the major difficulty with EEG data is the extraction of artifacts, for example due to eye movement, which

highly contaminates the raw data and lowers the signal power drastically with respect to the noise power.

Part II

Challenges in constrained tensor decompositions

Chapter 3

Constrained decompositions framework

This chapter contains numerous variations on the Tucker and CP decompositions presented in chapter 1, where factors have known properties. Each model is motivated by examples from data mining problems. Moreover, algorithms for constrained tensor decomposition are specifically designed. This chapter is inspired from the author's following publication [35] and is structured as follows:

- Different linear and non-linear constraint types are surveyed, along with associated constrained decomposition algorithms
- Algorithmic and theoretical issues related to compression under non-linear constraints are discussed.
- The use of dictionaries is studied through the spectral unmixing of the Snow data introduced in section 2.2.4.

3.1 Constraints as many faces of Tucker and CP decompositions

3.1.1 Motivation and general definition

Why constraints

A great strength of the CPD is its uniqueness under mild conditions always verified in practice. When first learning about this identifiability property of the CPD, one could believe that the CPD is the ultimate data mining model since for any real data set, estimates of factors obtained from computing the CPD will match with true underlying factors. As explained in section 1, things are not so simple. First, the multiway data have to follow approximately a low rank CP model, which is never really the case as discussed in chapter 2. Supposing this is true, and supposing the true rank is accurately guessed, optimization problem 1.50 may be quite difficult to solve depending on the amount of collinearity in factors, see 3.1.2. In any case since the CPD is computed from a noisy observation, estimation error is unavoidable and might be quite large if for instance Signal to Noise Ratio (SNR) is low, or columns of one factor are almost collinear.

However it is rare that no additional information is available on the factors of the CPD, especially if they bear some physical meaning. For example, in fluorescence spectroscopy, factors in the CPD stand for concentration profiles and spectra so that they are expected to be non-negative. This is tricky but even though factors can be uniquely recovered by computing the CPD, constraining the set of admissible solutions is absolutely necessary to ensure that the estimated factors are indeed consistent with the meaning of the CPD model in the application at hand. For example, suppose some estimated concentration profile in fluorescence spectroscopy is partially negative, then no physical meaning can be given to this profile. Probably such a negative factor is obtained because the data are noisy, but also because the low rank CPD model does not explain all the information in the data so that the best fitting factors $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ simply have no reason to stand

for emission, excitation spectra and concentration profiles. Therefore, imposing non-negativity constraints on this factor helps to interpret the results.

But constraints do not only make outputs of tensor decomposition models interpretable. As proved in [81], non-negativity constraints make the low rank CPD approximation problem well posed by restraining the set of admissible solutions. Moreover, Gorman *et al.* proved that constrained parameter estimation problems have better performance than their unconstrained counterparts if the true parameters satisfy the constraint [56]. In other words, if prior knowledge is available on the factors in the CPD or Tucker decomposition, it should be included inside the model, and constraints are actually one way to take this knowledge into account. Note that in some cases where data has very high SNR and the optimization problem is well-conditioned, imposing constraints has little to no impact on the estimation error on factors, but their use often come at the cost of some computation complexity. Thus the decomposition model to be used should be chosen carefully depending on the application of interest.

General constrained CPD

Definition 16 *Let $\boldsymbol{\mathcal{T}}$ a tensor following an approximate rank R CPD 1.47. An approximate constrained CPD of $\boldsymbol{\mathcal{T}}$ is the approximate CPD of $\boldsymbol{\mathcal{T}}$ where factors belong to known ensembles $\mathcal{S}_A$, $\mathcal{S}_B$ and $\mathcal{S}_C$:*

$$\begin{cases} \boldsymbol{\mathcal{T}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\,\boldsymbol{\Sigma}_R + \boldsymbol{\mathcal{E}} \\ \boldsymbol{A} \in \mathcal{S}_A, \boldsymbol{B} \in \mathcal{S}_B, \boldsymbol{C} \in \mathcal{S}_C \end{cases} \tag{3.1}$$

where distribution of random variable $\boldsymbol{\mathcal{E}}$ is known.

We refer to the approximate constrained CPD simply as constrained CPD, but constraining the exact CPD can be useful even when the decomposition is unique, see [106, 107]. In what follows and for the sake of simplicity, only $\boldsymbol{C}$ is considered constrained. Generalization to models where all factors are constrained is straightforward.

In the following, various constraint sets $\mathcal{S}_C$ are defined and properties of these various instances of constrained CPD are explored. An important aspect of the constrained CPD is that the constraint space needs to give good information on the solution to be obtained, since the constraint is imposed rigorously in the models. In other words, the *a priori* information on $\boldsymbol{C}$ must yield a consistent estimator of $\boldsymbol{C}$ through constrained CPD otherwise model (3.1) is simply wrong. For instance, if true factors are non-negative, imposing a non-positivity constraint will (of course) decrease estimation performances and interpretability. On the other hand, true underlying sources may exist in reality, like spectra in spectroscopy, but may not be exactly the factors in the mean of an approximate CP model since the decomposition models are often somewhat inexact in practice. In that case the approximate CPD yields a biased estimator of underlying sources, and constraints help to reduce the research space around the true underlying sources provided they are satisfied by these sources. A toy example summarizing these issues is given below.

Example 3 *Suppose a scalar datum x is observed without noise, stemming from an underlying parameter y so that $y^3 = x$ and $y > 0$. In most cases the model linking y to x is not known so that an a priori model is used to estimate y from x, say $\hat{y}^2 = x$. The parameter $\hat{y}$ of this a priori model is not identifiable since both $\hat{y} = \sqrt{x}$ and $\hat{y} = -\sqrt{x}$ are solutions. Moreover, the estimator $\hat{y} = \sqrt{x}$ is not a consistent estimator of y since the true model is $y^3 = x$.*

Constraints can be included in the a priori model with various effects. First, one may know that the true underlying parameter y is positive. Then including a non-negativity constraint $\hat{y} \geqslant 0$ in the a priori model yields identifiable parameters in the a priori model, but does not decrease the estimation error. Another kind of constraint on $\hat{y}$ can be for example $\hat{y} \in \mathbb{N}$, which is a bad constraint in the sense that y may not be an integer. Moreover, the set of parameters satisfying both the constraint and the a priori model is empty. A third possible constraint may be $\hat{y}^3 = x$, which is not helpfull here since that means the true model is known. But still if the a priori model is constrained by this equation, the set of admissible solutions is also empty (except if x is 1 or 0).

In the constrained tensor decomposition scenario, the model may also be inexact, and the constraints are hopefully well matched to the true underlying sources. However it may happen that even through underlying parameters satisfy the constraints, the CPD is simply a loose model for extracting the underlying sources and final estimates of the sources will be inaccurate, just like the last constraint type in the previous example. This means that model design is complex

and it is not sufficient to input many constraints in the optimization algorithm to obtain a good final estimation of the underlying sources. Rather, a precise understanding of the physical laws governing the problem at hand is crucial, since *a posteriori* it is not always possible to check whether the constraints were well designed or not. To conclude, if the low rank approximate CPD model provides a consistent estimator of the underlying sources, then additional constraints satisfied by the underlying sources can be used to design better models than plain approximate CPD.

3.1.2 Linear constraints

A first class of meaningful constraints are linear constraints, *i.e.* when $\boldsymbol{C}$ belongs to some linear subspace of $\mathbb{R}^{M\times R}$. Linear constraints are often used for compressing multiway arrays. However linear constraints also cover more subtle problems like collinearity among columns [19] or known spanning families of the sources, which are typically discretized functions from a special subclass of continuous mappings (B-splines, harmonics, wavelets...) [112]. Since factor $\boldsymbol{C}$ is a matrix, linear constraints on $\boldsymbol{C}$ are of the form $\boldsymbol{C} = \sum_i \boldsymbol{W}_i \boldsymbol{C}_c^i \boldsymbol{H}_i$, where $\boldsymbol{W}_i$ and $\boldsymbol{H}_i$ are linear operators on the columns and rows of $\boldsymbol{C}$, and $\boldsymbol{C}_c^i$ are score matrices in a feature space[1]. Rank-one linear operators of the form $\boldsymbol{C} = \boldsymbol{W}\boldsymbol{C}_c\boldsymbol{H}$ are particular interest to express changes of basis and collinearity among columns. In what follows, we first study how to handle $\boldsymbol{W}$, before dealing with $\boldsymbol{H}$. The first formalization of linearly constrained CPD dates back to Caroll and was called CANDELINC [26].

Known basis for CPD factors

Let $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{K\times L\times M}$ and consider the following linearly constrained CPD with small tensor rank R:

$$\begin{cases} \boldsymbol{\mathcal{T}} = (\boldsymbol{A}\otimes\boldsymbol{B}\otimes\boldsymbol{C})\,\boldsymbol{\Sigma}_R + \boldsymbol{\mathcal{E}} \\ \boldsymbol{C} = \boldsymbol{W}\boldsymbol{C}_c \end{cases} \tag{3.2}$$

for some matrix $\boldsymbol{W}$ and scores $\boldsymbol{C}_c$. $\boldsymbol{W}$ and $\boldsymbol{C}_c$ can be known or unknown. Here columns of $\boldsymbol{C}$ are constrained to live in the span of columns of $\boldsymbol{W}$.

Since $\boldsymbol{C}$ is a M by R matrix, columns of $\boldsymbol{W}$ have to span a linear subspace of dimension smaller than M. Indeed if $\boldsymbol{W}$ has more than M free columns, the span of columns of $\boldsymbol{W}$ is the whole definition space of $\boldsymbol{C}$, and (3.2) is equivalent to an unconstrained approximate CPD. From now on we suppose that $\boldsymbol{W}$ has column rank lower than M. Moreover, if $\boldsymbol{W}$ is not full column rank, a smaller matrix $\boldsymbol{W}'$ can be obtained by removing redundancy in columns of $\boldsymbol{W}$. Thus we also suppose that $\boldsymbol{W}$ is full column rank.

Matrix $\boldsymbol{W}$ provides with a basis of representation for columns of factor $\boldsymbol{C}$. Let us give a toy example. If columns of $\boldsymbol{C}$ are spanned by sinusoidal signals, say

$$C_{jr} = \sum_{i=1}^{5} \lambda_{ir} sin(ij\pi/10), \tag{3.3}$$

then each $\boldsymbol{c}_r = \boldsymbol{C}_{:r}$ is expressed by coefficients λ_r in the basis

$$\boldsymbol{W} = \begin{bmatrix} sin(\pi/10) & \dots & sin(\pi/2) \\ sin(\pi/5) & \dots & sin(\pi) \\ & \vdots & \\ sin(\pi/2) & \dots & sin(\pi/2) \end{bmatrix}. \tag{3.4}$$

In this toy example, since all columns of $\boldsymbol{C}$ are expressed in the span of $\boldsymbol{W}$ with only five coefficients each, $\boldsymbol{C}_c$ is a small matrix $\mathbb{R}^{5\times R}$.

Another example of known basis of representation can be found in the chemometrics literature [112]. Since factors stand for emission and excitation spectra which are known to be smooth for used sampling rates for some chemicals, Timmerman et. al. proposed to impose this smoothness on $\boldsymbol{C}$ by using a basis of discretized B-spline functions $\boldsymbol{W}$. A B-spline function is a polynomial function of given degree which is constrained to vanish outside a given interval. Since $\boldsymbol{C}$ is smooth, the working hypothesis is that it can be represented by a few spline functions.

[1] From proposition 2, all linear maps on matrices are Kronecker products of linear maps on the columns and rows of matrices, and $(\boldsymbol{W}\boxtimes\boldsymbol{V})\,\boldsymbol{C}_c = \boldsymbol{W}\boldsymbol{C}_c\boldsymbol{V}^{\mathsf{T}}$.

Compression

When a basis of representation $\boldsymbol{W}$ for factor $\boldsymbol{C}$ is known, $\boldsymbol{W}$ can be used to preprocess the tensor data so that a smaller core tensor $\boldsymbol{\mathcal{G}}$ can be decomposed instead of $\boldsymbol{\mathcal{T}}$. Indeed, (3.2) can be rewritten as

$$\boldsymbol{\mathcal{T}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{W}\boldsymbol{C}_c)\, \boldsymbol{\Sigma}_R + \boldsymbol{\mathcal{E}} \tag{3.5}$$

and since $\boldsymbol{W}$ is full column rank, it admits a left inverse $\boldsymbol{W}^\dagger = \left(\boldsymbol{W}^T\boldsymbol{W}\right)^{-1}\boldsymbol{W}^\mathsf{T}$ so that under white Gaussian noise,

$$\begin{aligned}\left(\boldsymbol{I} \otimes \boldsymbol{I} \otimes \boldsymbol{W}^\dagger\right)\boldsymbol{\mathcal{T}} &= (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C}_c)\, \boldsymbol{\Sigma}_R + \left(\boldsymbol{I} \otimes \boldsymbol{I} \otimes \boldsymbol{W}^\dagger\right)\boldsymbol{\mathcal{E}} \\ &= \boldsymbol{\mathcal{G}} + \boldsymbol{\mathcal{E}}'\end{aligned} \tag{3.6}$$

where $\boldsymbol{\mathcal{E}}' \sim \mathcal{AN}\left(\boldsymbol{0}, \boldsymbol{I} \otimes \boldsymbol{I} \otimes \left(\boldsymbol{W}^\mathsf{T}\boldsymbol{W}\right)^{-1}\right)$. If $\boldsymbol{W}$ is in $\mathbb{R}^{M \times R_1}$, $\boldsymbol{\mathcal{G}}$ belongs to $\mathbb{R}^K \otimes \mathbb{R}^L \otimes \mathbb{R}^{R_1}$ and because R_1 is necessarily smaller than R, $\boldsymbol{\mathcal{G}}$ is smaller than $\boldsymbol{\mathcal{T}}$. Moreover, $\boldsymbol{\mathcal{G}}$ can be used as a new data tensor to perform an approximate CPD with known noise covariance. The compression scheme is summarized in Figure 3.1.

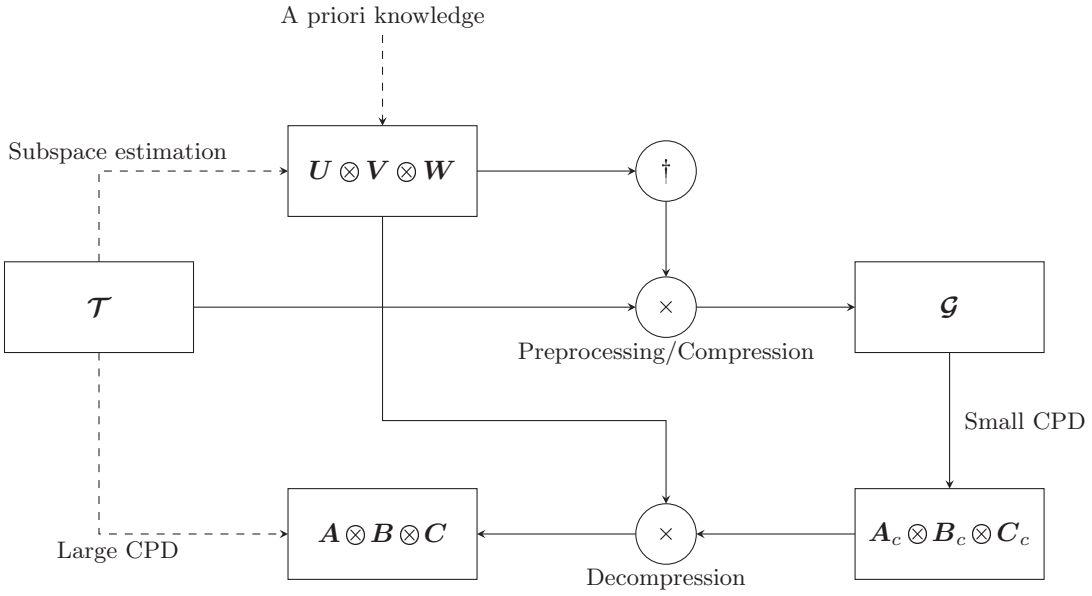


Figure 3.1: Compression scheme (noiseless)

A case of linear constraints often encountered is when $\boldsymbol{W}$ is orthonormal. Then the noise covariance after preprocessing in (3.6) remains white, so that a whitening step after compression is unnecessary. It is possible to learn an orthonormal matrix $\boldsymbol{W}$ from the data tensor $\boldsymbol{\mathcal{T}}$ as discussed below.

Collinear columns in factors

Let us now consider linear constraints acting on the right of matrix $\boldsymbol{C}$, *i.e.* on the rows of $\boldsymbol{C}$. The following linearly constrained model is obtained:

$$\begin{cases} \boldsymbol{\mathcal{T}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\, \boldsymbol{\Sigma}_R + \boldsymbol{\mathcal{E}} \\ \boldsymbol{C} = \boldsymbol{C}_c\boldsymbol{H} \end{cases}. \tag{3.7}$$

As discussed in the previous paragraph, $\boldsymbol{H}$ can be understood as a basis of rows of $\boldsymbol{C}$, but this is not the usual interpretation of (3.7). Rather it is often understood as a way to handle correlation in columns of factor $\boldsymbol{C}$. Indeed, for (3.7) to be a constrained decomposition model, the span of rows of $\boldsymbol{H}$ needs to be a strict subspace of $\mathbb{R}^R$. Again, rows of $\boldsymbol{H}$ can be chosen independent otherwise another constraint matrix $\boldsymbol{H}'$ can be obtained from $\boldsymbol{H}$ by removing redundancy. For $\boldsymbol{H}$

of size $R_1 \times R$, R_1 is strictly smaller than R. Thus each column of $\boldsymbol{C}$ is a linear combination of columns of $\boldsymbol{C}_c$ with strictly less than R coefficients in $\boldsymbol{H}$, *i.e.* $\boldsymbol{C}_c$ is a processed version of factor $\boldsymbol{C}$ where collinearity is removed.

Model (3.7) was simultaneously suggested in [19] and [42, 44], respectively named the PARALIND model and $(L_r, L_r, 1)$ block term decomposition. We choose to follow notations from PARALIND since they fit better with our formalism, but the block term decomposition benefits from generalization properties to other block terms models, which are not discussed in this manuscript [42]. A toy example of tensor decomposition where collinear columns can be handled through PARALIND is given below.

Example 4 *Let us build an experiment where factors should have different column ranks. Take a solution with three fluorescent components at different concentrations. Using a spectrophotometer, a mixture of the three emission spectra and the three excitation spectra can be acquired in the form of a matrix. Now by adding a bit of the third component and diluting the whole solution, running the experiment again gives another matrix. Repeating this several times will result in a collection of matrices, our measurement tensor, where the two first concentration profiles are collinear. Thus the column rank of the factor $\boldsymbol{C}$ related to concentration will be two, even though the rank of the tensor will be three. A better model would thus use a reduced set $\boldsymbol{C_c}$ of parameters by setting*

$$\boldsymbol{C} = \boldsymbol{C_c} \begin{bmatrix} 1 & \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where λ is the ratio between the two parallel concentration profiles. This model is exactly the PARALIND model discussed above.

PARALIND provides a modeling response to the problem of collinear columns in CPD factors. Indeed, assume a factor $\boldsymbol{C}$ has two colinear columns. Running an ALS algorithm as described in Algorithm 1 is bound to fail for two reasons. First, the conditioning of the linear system with respect to $\boldsymbol{A}$ and $\boldsymbol{B}$ is poor since it depends on the conditioning of $\boldsymbol{C}$. Second, as explained below, factors $\boldsymbol{A}$ and $\boldsymbol{B}$ are identifiable only up to rotations among some columns.

A significant issue with PARALIND is that $\boldsymbol{H}$ is not identifiable, *i.e.* multiple pairs $(\boldsymbol{H}, \boldsymbol{C}_c)$ yield the same $\boldsymbol{C}$. This issue is adressed in [24]. Also, there can be an inherent ambiguity among factors $\boldsymbol{A}$ and $\boldsymbol{B}$. Indeed, assume that the two first columns of $\boldsymbol{C}$ are collinear *i.e.* $\boldsymbol{c}_1 = \boldsymbol{c}_2$. Then the following holds:

$$\begin{aligned} \boldsymbol{\mathcal{T}} &= \sum_{r=1}^{R} \boldsymbol{a}_r \otimes \boldsymbol{b}_r \otimes \boldsymbol{c}_r \\ &= (\boldsymbol{a}_1 \otimes \boldsymbol{b}_1 + \boldsymbol{a}_2 \otimes \boldsymbol{b}_2) \otimes \boldsymbol{c}_1 + \sum_{r=3}^{R} \boldsymbol{a}_r \otimes \boldsymbol{b}_r \otimes \boldsymbol{c}_r \\ &= \boldsymbol{A}_1 \boldsymbol{B}_1^{\mathsf{T}} \otimes \boldsymbol{c}_1 + \sum_{r=3}^{R} \boldsymbol{a}_r \otimes \boldsymbol{b}_r \otimes \boldsymbol{c}_r \\ &= \boldsymbol{A}_1 \boldsymbol{P} \boldsymbol{P}^{-1} \boldsymbol{B}_1^{\mathsf{T}} \otimes \boldsymbol{c}_1 + \sum_{r=3}^{R} \boldsymbol{a}_r \otimes \boldsymbol{b}_r \otimes \boldsymbol{c}_r \end{aligned} \tag{3.8}$$

for any non-singular matrix $\boldsymbol{P}$ and $\boldsymbol{A}_1 = [\boldsymbol{a}_1,\ \boldsymbol{a}_2]$, $\boldsymbol{B}_1 = [\boldsymbol{b}_1,\ \boldsymbol{b}_2]$.

In a more general setting where columns are not collinear two by two, the loss of identifiability is harder to characterize. Bounds on the identifiability depending on the so-called Kruskal rank can be found in [77] while tighter bounds are provided in [49]. A matrix has a Kruskal rank k if any subset of columns is full rank, and if this does not hold for $k+1$. By construction, Kruskal rank cannot exceed matrix rank. For instance in the above example (3.8), factor matrix $\boldsymbol{C}$ has Kruskal rank 2, and therefore does not satisfy the Kruskal bound for identifiability. Further work on identifiability of the PARALIND model itself are due to De Almeida *et. al.* [108] and De Lathauwer *et. al.* [42, 46, 104].

Another major issue with PARALIND is that there is no simple way to estimate $\boldsymbol{H}$ from the data without prior knowledge. This is to be opposed to the linear constraint on the left of $\boldsymbol{C}$ where $\boldsymbol{W}$ can be estimated as described later in section 3.1.2. Thus a two-step estimation procedure where $\boldsymbol{H}$ is first estimated from the data before decomposing the tensor is not generically possible. Moreover, even if $\boldsymbol{H}$ is known, a preprocessing of the data to obtain an unconstrained CPD is also not possible because $\boldsymbol{H}$ acts on the right of $\boldsymbol{C}$ and is therefore contracted on both $\boldsymbol{\Sigma}$ and $\boldsymbol{C}_c$ in the CPD. Thus specific algorithms have to be designed to compute (3.7). Algorithm 3 provides a possible implementation based on ALS, which originates from [19] and [44].

Algorithm 3 An ALS algorithm for PARALIND.

Given $\boldsymbol{\mathcal{T}}$ and initial factors $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$,
while convergence criterion is not met **do**

$$\begin{aligned}
\boldsymbol{A} &= \boldsymbol{\mathcal{T}}_{(1)} \left(\boldsymbol{B} \odot \boldsymbol{C}\right) \left(\boldsymbol{B}^{\mathsf{T}}\boldsymbol{B} \boxdot \boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}\right)^{-1} \\
\boldsymbol{B} &= \boldsymbol{\mathcal{T}}_{(2)} \left(\boldsymbol{A} \odot \boldsymbol{C}\right) \left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} \boxdot \boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}\right)^{-1} \\
\boldsymbol{C}_c &= \boldsymbol{\mathcal{T}}_{(3)} \left(\boldsymbol{A} \odot \boldsymbol{B}\right) \boldsymbol{H}^{\mathsf{T}} \left[\boldsymbol{H} \left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} \boxdot \boldsymbol{B}^{\mathsf{T}}\boldsymbol{B}\right) \boldsymbol{H}^{T}\right]^{-1} \\
\boldsymbol{H} &= \boldsymbol{C}_c^{\dagger}\boldsymbol{\mathcal{T}}_{(3)} \left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} \boxdot \boldsymbol{B}^{\mathsf{T}}\boldsymbol{B}\right)^{-1} \\
\boldsymbol{C} &= \boldsymbol{C}_c\boldsymbol{H}
\end{aligned}$$

end while

Low CPD rank induces compression and collinear columns

When decomposing a tensor using an approximate CP decomposition, as explained in section 1.4.1, the CPD model is low rank. That is, some factors in the CPD have more rows than columns. Thus rows of these factors will be linearly dependent, since for matrices the rank is always smaller than both dimensions. In other words, under the assumption that the rank is smaller than all dimensions of the tensor, it is always possible to find tall matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ so that $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{A}_c$, $\boldsymbol{B} = \boldsymbol{V}\boldsymbol{B}_c$ and $\boldsymbol{C} = \boldsymbol{W}\boldsymbol{C}_c$.

Since low tensor rank hypothesis in the approximate CP models leads to collinearity in the rows of factors, compression of the data tensor is always possible in this context. If matrix $\boldsymbol{W}$ is exactly known, then compression as described above can be applied on $\boldsymbol{C}$ without loss of information on any factor. However, when only the data tensor is provided, which is usually the case, the column space of factor $\boldsymbol{C}$ can be learned through the HOSVD of the tensor. If the SNR is high, Algorithm 2 provides a fairly well estimated basis $\boldsymbol{W}$ and the compression does not significantly reduce the amount of information contained on $\boldsymbol{C}$ in the data [45]. Other subspace estimation methods can be used in theory like the QR decomposition. However, because the approximate nature of the CPD imposes a strict selection of a subspace of the estimated column span, a ranking of the importance of each basis vector is mandatory.

More generally, a low-rank constraint can be cast on factor $\boldsymbol{C}$:

$$\mathcal{S}_C = \{\boldsymbol{X} \in \mathbb{R}^{M \times R} \mid \text{rank}\{\boldsymbol{X}\} < \min(M, R)\}. \tag{3.9}$$

Here the rank of $\boldsymbol{C}$ is strictly smaller than both dimensions. Thus not only rows but also columns of $\boldsymbol{C}$ are linearly dependent. $\mathcal{S}_C$ can be linearized around $\boldsymbol{C}$ given column and row bases $\boldsymbol{U}$ and $\boldsymbol{V}$ of $\boldsymbol{C} = \boldsymbol{U}\boldsymbol{C}_c\boldsymbol{V}$. Therefore, a rank constraint can be seen as two linear constraints on both the right and left of $\boldsymbol{C}$, which means that compression and collinear columns should both be considered.

3.1.3 Non linear constraints

Non-negativity

Many applications require nonlinear constraints. One frequently encountered is non-negativity, which is essential in chemometrics or hyperspectral imaging, see chapter 2. With a non-negativity constraint on the third mode, the constrained CPD becomes

$$\begin{cases} \boldsymbol{\mathcal{T}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\boldsymbol{\mathcal{I}}_R + \boldsymbol{\mathcal{E}} \\ \boldsymbol{C} \in \mathbb{R}_+^{M \times R} \end{cases}. \tag{3.10}$$

This has been used extensively whenever factors bear physical interpretation in chemometrics [16]. It is also especially popular for two-way arrays under the Non-negative Matrix Factorization (NMF) acronym [79], since this constraint for matrices often restores identifiability of the factors.

For higher order tensors, non-negativity constraints ensure that a best low rank approximation exists [81] for all norms. This is not the only constrained CPD for which the problem of existence of a best low rank approximation is solved. Indeed, it is sufficient that the set of constrained rank R tensors $\left\{\boldsymbol{\mathcal{T}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\boldsymbol{\mathcal{I}}_R \in \mathbb{R}^K \otimes \mathbb{R}^L \otimes \mathbb{R}^M | \boldsymbol{A} \in \mathcal{S}_A,\ \boldsymbol{B} \in \mathcal{S}_B,\ \boldsymbol{C} \in \mathcal{S}_C\right\}$ is closed, since closure prevents parameters to cancel out by growing to infinity. However in practice few research

on the well-posedness of approximate decompositions has been made for other constraints than non-negativity, some references for orthogonality constraints are [75, 107].

An efficient and simple algorithm for computing non-negative CP decomposition can be built from the ALS. The so-called Alternating Non-negative Least Squares (ANLS) simply projects the solution of the linear system over $\boldsymbol{C}$ on the non-negative orthant, see Algorithm 4. Below we show that this can also be seen as an instance of a proximal gradient algorithm used in a block-coordinate fashion.

Algorithm 4 ANLS / Alternate Outerloop Proximal Gradient algorithm for non-negative CP decomposition.

Given $\boldsymbol{\mathcal{T}}$ and initial factors $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$,
while convergence criterion is not met **do**

$$\begin{aligned}
\boldsymbol{A} &= \boldsymbol{\mathcal{T}}_{(1)} \left(\boldsymbol{B} \odot \boldsymbol{C}\right) \left(\boldsymbol{B}^{\mathsf{T}}\boldsymbol{B} \boxdot \boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}\right)^{-1} \\
\boldsymbol{B} &= \boldsymbol{\mathcal{T}}_{(2)} \left(\boldsymbol{A} \odot \boldsymbol{C}\right) \left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} \boxdot \boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}\right)^{-1} \\
\boldsymbol{C} &= \left[\boldsymbol{\mathcal{T}}_{(3)} \left(\boldsymbol{A} \odot \boldsymbol{B}\right) \left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} \boxdot \boldsymbol{B}^{\mathsf{T}}\boldsymbol{B}\right)^{-1}\right]^{+}
\end{aligned}$$

end while

Example 5 *The proximal gradient method for solving constrained optimization problems is closely related to traditional projected gradient descent methods. Indeed, it is an iterative algorithm that requires to compute the gradient at any given point of the unconstrained cost function, and then projects the new estimate on the constraint space. However, to ensure both convergence and satisfied constraints, the projection on the constraint space is done using a particular operator called the proximal operator. It is easy to prove that projected gradient and proximal gradient are equivalent if an orthogonal projector on the constraint space is known.*

For solving the non-negative CPD optimization problem

$$\begin{array}{ll} \underset{\boldsymbol{C}}{\operatorname{argmin}} & \|\boldsymbol{\mathcal{T}} - (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\boldsymbol{\mathcal{I}}_R\|_F^2 \\ s.t. & \boldsymbol{C} \in \mathbb{R}_+^{K \times R}. \end{array} \tag{3.11}$$

the proximal gradient can be used in a block-coordinate or ALS spirit by computing the gradient and proximal operators for each factor sequentially. Since (3.11) *is linear with respect to each factor, a gradient method with optimal (matrix) step is equivalent to the least squares update, so that without projections, estimates of factors $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are estimated sequentially as in traditional ALS.*

That leaves the computation of the proximal operator of the non-negativity constraint on $\boldsymbol{C}$. By definition, the proximal operator Π_λ in this case is given by

$$\Pi_\lambda(\boldsymbol{X}) = \underset{\boldsymbol{U}}{\operatorname{argmin}}\ \eta(\boldsymbol{U}) + \lambda\|\boldsymbol{X} - \boldsymbol{U}\|_2^2 \tag{3.12}$$

where $\eta(\boldsymbol{X})$ is the characteristic function of the set of matrices with non-negative coefficients, i.e.

$$\eta(\boldsymbol{U}) = \begin{cases} 0 & \text{if } U_{ij} \geqslant 0 \quad \forall U_{ij} \\ +\infty & \text{if there exist } U_{ij} < 0 \end{cases}. \tag{3.13}$$

It can be seen that $\Pi_\lambda(\boldsymbol{X}) = [\boldsymbol{X}]^+$ for all $\boldsymbol{X}, \lambda$, which means that all negative values in least squares estimate of $\boldsymbol{C}$ are set to zero while leaving the other values intact. This shows that an alternate outerloop of proximal gradient amounts to the ANLS from [28] recalled in Algorithm 4.

A major issue with ANLS is however that there are no guarantee of convergence of the algorithm, since the cost function is not necessarily decreasing at each iteration. This is due to the alternate procedure of least squares estimate and projection on the non-negative orthant. To improve convergence properties of ANLS, it is possible to resort to non-negative least squares updates [78] alternated on each factor. An efficient implementation of this quadratic programming approach is due to Bro *et. al.* [18] and is called Fast Non-Negative Least Squares (FFNLS). Other

works have focused on applying Alternating Direction Method of Multipliers (ADMM) [68] again alternating on each factor, or using Gauss-Newton second-order methods sped up by conjugate gradient [120].

Non-negativity constraints can also be used to define the following constrained approximate Tucker model:

$$\begin{cases} \boldsymbol{\mathcal{T}} = (\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\, \boldsymbol{\mathcal{G}} + \boldsymbol{\mathcal{E}} \\ \boldsymbol{W} \in \mathbb{R}_{+}^{M \times R_1} \end{cases} . \tag{3.14}$$

This model may be used to find compression matrices that are non-negative so as to avoid the problem evoked below in section 3.2. However, computing the non-negative Tucker decomposition is more costly than computing the unconstrained HOSVD through three SVDs. As a consequence non-negative Tucker may have trouble to handle large data sets. On the other hand, if non-negativity constraints are imposed on all modes, the model may be identifiable. Remember that the lack of identifiability is a major drawback of Tucker decomposition, an issue which can here be fixed using constraints.

Sparsity

Another common non-linear constraint is the ℓ_0 sparsity constraint [23, 82] :

$$\begin{cases} \boldsymbol{\mathcal{T}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\, \boldsymbol{\mathcal{I}}_R + \boldsymbol{\mathcal{E}} \\ \ell_0(\boldsymbol{C}) \leqslant \delta \end{cases} \tag{3.15}$$

for some integer δ and the ℓ_0 *pseudo-norm* can be taken rowwise, columnwise or on all elements. This means that each of the R components in the CPD contributes only sparsely to tensor $\boldsymbol{\mathcal{T}}$ on the third mode. Sparsity constraints are meaningful when at least one of the factor is understood as a dictionary, and factor $\boldsymbol{C}$ has a small number of non-zero coefficients. To our knowledge, the impact of sparsity constraints on the existence and uniqueness of a best approximate CPD has not been studied yet. The question that should be tackled in future works is the following: what are the conditions for which the set

$$\left\{ \boldsymbol{\mathcal{T}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\, \boldsymbol{\mathcal{I}}_R \in \mathbb{R}^K \otimes \mathbb{R}^L \otimes \mathbb{R}^M \mid \ell_0(\boldsymbol{A}) \leqslant \delta_A,\ \ell_0(\boldsymbol{B}) \leqslant \delta_B,\ \ell_0(\boldsymbol{C}) \leqslant \delta_C \right\} \tag{3.16}$$

is closed ?

The ℓ_1 norm can be used in a similar fashion, as a relaxation of the ℓ_0 norm for the optimization routine, or as a constraint in itself. For instance, for $\boldsymbol{C}$ stemming from the decomposition of hyperspectral data as presented in section 2.2.4, rows of $\boldsymbol{C}$ should sum to one which leads to $\ell_1(\boldsymbol{C}) = 1$ rowwise. The ℓ_1 norm also serves as a regularizer for dictionary learning with matrices. Sparsity for dictionary-based CP decomposition is explored in the next section.

To design a simple algorithm for computing (3.15), ALS can once again be modified by projecting the unconstrained linear estimate in each step onto the constraint space. The projection on factors of given ℓ_0 pseudo-norm and ℓ_1 norm are computed by the two following operators:

$$\begin{aligned} \Pi_{\ell_0,\delta}(\boldsymbol{X}) = &\ \boldsymbol{Y} \text{ where } \boldsymbol{Y} \text{ contains only zeros and the } \delta \text{ greater values of } \boldsymbol{X} \\ \Pi_{\ell_1,\delta}(\boldsymbol{X}) = &\begin{cases} x - \delta & \text{if } x \geqslant \delta \\ 0 & \text{if } -\delta < x < \delta \quad \forall x = X_{ij} \\ x + \delta & \text{otherwise} \end{cases} \end{aligned} . \tag{3.17}$$

If the sparsity constraint is an equality constraint with non-negativity futher imposed, *i.e.* each row of factor $\boldsymbol{C}$ should sum to 1 exactly and be non-negative, then the constraint space is the probability simplex and the solution of the projection is given by the solution of the water filling problem [13, 121]. A simple implementation is given by the following pseudo-code from [121]:

A sparse Tucker decomposition can be defined as:

$$\begin{cases} \boldsymbol{\mathcal{T}} = (\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\, \boldsymbol{\mathcal{G}} + \boldsymbol{\mathcal{E}} \\ \ell_0\left(\boldsymbol{\mathcal{G}}\right) \leqslant \delta \end{cases} \tag{3.18}$$

which became more popular in the recent years within the dictionary learning community [6, 23]. Even though restoring identifiability is the main advantage of imposing sparsity on the core of the Tucker decomposition, no discussion on identifiability of this model can be found in the literature.

Algorithm 5 Euclidean projection of a vector onto the probability simplex

INPUT: vector $\boldsymbol{y} \in \mathbb{R}^N$,

1. Sort $\boldsymbol{y}$ into $\boldsymbol{u} : u_1 \geqslant \cdots \geqslant u_N$
2. Find $\rho = \max \{1 \leqslant j \leqslant N \mid 1 - \sum_{i=1}^{j} (u_i - u_j) > 0\}$
3. Define $\lambda = \frac{1}{\rho} \left(1 - \sum_{i=1}^{\rho} u_i\right)$

OUTPUT: $\boldsymbol{x}$ s.t. $x_i = \max(y_i + \lambda, 0)$

3.2 Compression under non-linear constraints

In the previous section, a method for compressing low rank multiway arrays of data was described. We stress that compression has many benefits when used as a preprocessing tool for computing the CPD. First it reduces memory requirements of computing the CPD since only small factors and a small core tensor are used. However learning the good basis for the compression may be itself quite costly in terms of memory. Second, on the computation speed side, since the result of compression through HOSVD is deterministic, it only needs to be run once. On the other hand, any algorithm for computing the CPD requires multiple initializations. This means that even if compression is a lengthy process, when using the CPD on the small core, the gain in computation speed is multiplied by the number of times the CPD algorithm is used. Finally, the loss of accuracy is usually negligible if the compression basis is well estimated, which is typically true for Gaussian noise and high SNR [45].

If however constraints are imposed on some factors in the CPD of a tensor $\boldsymbol{\mathcal{T}}$, splitting the compression and decomposition steps into two separate process becomes difficult since constraints are applied initially in the uncompressed space. To propagate constraints directly on the compressed space and keep split compression and decomposition processes, it is necessary to design a compression method accounting for these additional constraints. This has two major consequences. First, the compression method can no longer be the HOSVD, and typically a compression algorithm accounting for non-linear constraints will be more costly than three SVDs. Second, compression has to be computed for every combination of constraints that are applied on the tensor. Indeed, it is common in data science to try many different models, and check *a posteriori* which one provides the best results.

Therefore in this section an alternative method using unconstrained compression is presented, accounting for constraints only when computing the CPD of the small core tensor. This method is described in the following published paper [32], where two algorithms were proposed. Only the algorithm based on ALS is presented here since it is the simplest one and the most effective. After describing the problem formally and showing a few failed attempts at solving it efficiently, algorithm PROCO-ALS is detailed. It is shown than any constraint space for which a projector is known can be handled with PROCO-ALS. Finally, following the ideas developed in PROCO-ALS, primal-dual proximal algorithms are designed to tackle the compression under constraint problem, with half-hearted results with respect to PROCO-ALS.

3.2.1 Introduction to compression under constraints

Problem statement

Let $\boldsymbol{\mathcal{T}}$ be a tensor that is to be decomposed using the non-linearly constrained CPD model $\boldsymbol{C} \in \mathcal{S}_C$. Let $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ be three orthonormal basis of columns of factors $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$. Then with notations from section 3.1, constrained decomposition (3.1) can be cast equivalently in a compressed domain:

$$\begin{cases} \boldsymbol{\mathcal{G}} = (\boldsymbol{A}_c \otimes \boldsymbol{B}_c \otimes \boldsymbol{C}_c)\boldsymbol{\mathcal{I}}_R + \boldsymbol{\mathcal{E}}_c \\ \boldsymbol{W}\boldsymbol{C}_c \in \mathcal{S}_C \end{cases} \tag{3.19}$$

where $\boldsymbol{\mathcal{E}}_c = \left(\boldsymbol{I} \otimes \boldsymbol{I} \otimes \boldsymbol{W}^{\mathsf{T}}\right) \boldsymbol{\mathcal{E}}$ still has variance $\sigma^2 \boldsymbol{I}^{\otimes 3}$.

The compressed space is defined as the subspace containing $\boldsymbol{C}_c$ whereas the uncompressed space contains $\boldsymbol{C}$. In (3.19), constraints are applied in the uncompressed space whereas the CPD is done in the compressed domain. This poses a serious issue when running a decomposition algorithm

directly in the compressed space, since there is no simple way to compute the projection of $\boldsymbol{C}_c$ onto $\boldsymbol{W}^\mathsf{T}(\mathcal{S}_C)$ even if projection on $\mathcal{S}_C$ is obvious. This point is detailed in what follows.

A good working example for understanding the intrinsic issues of solving (3.19) is the non-negative CPD. It was originally the starting point of the original research described in this section, although it is now generalized to many other constraint types.

Projecting on the set of non-negative factors is simple. Given a matrix $\boldsymbol{X}$, the closest matrix $\boldsymbol{Y}$ with coefficients in the non-negative orthant is obtained by taking the positive part[2] of all coefficients in $\boldsymbol{X}$

$$Y_{ij} = \max(X_{ij}, 0) \quad \equiv \quad \boldsymbol{Y} = [\boldsymbol{X}]^+ . \tag{3.20}$$

However if compression is used, the transformed set of constraints applied to $\boldsymbol{C}_c$ is defined by $\boldsymbol{W}^\mathsf{T}(\mathcal{S}_c) = \{\boldsymbol{X} \in \mathbb{R}^{K\times R} \mid \boldsymbol{W}\boldsymbol{X} \geq 0\}$ where $\geq$ is the coefficient-wise comparison. If $\boldsymbol{W}^\mathsf{T}$ were full rank orthonormal, then a projection $\boldsymbol{Y}$ on $\boldsymbol{W}^\mathsf{T}(\mathcal{S}_c)$ of $\boldsymbol{X}$ would be given by

$$\boldsymbol{Y} = \Pi_{\boldsymbol{W}(\mathcal{S}_C)}(\boldsymbol{X}) := \boldsymbol{W}^\mathsf{T}[\boldsymbol{W}\boldsymbol{X}]^+ \tag{3.21}$$

since $\boldsymbol{W}$ would be an isomorphism from the non-negative orthant to the transformed constraint space. Moreover if $\boldsymbol{W}^\mathsf{T}$ is full column rank, then similarly the projection on the transformed constraints can be performed by applying (3.21). Indeed, if $\boldsymbol{W}\boldsymbol{W}^\mathsf{T} = \boldsymbol{I}$, then $\Pi^2_{\boldsymbol{W}(\mathcal{S}_C)} = \Pi_{\boldsymbol{W}(\mathcal{S}_C)}$ which defines a projector. But when $\boldsymbol{W}$ is obtained through HOSVD of the tensor, it is always row-rank deficient, that is $\boldsymbol{W}\boldsymbol{W}^\mathsf{T} \neq \boldsymbol{I}$, even though $\boldsymbol{W}^\mathsf{T}\boldsymbol{W} = \boldsymbol{I}$. To obtain a projector from (3.21) in the general case, $[\boldsymbol{W}\boldsymbol{X}]^+$ has to live in the span of columns of $\boldsymbol{W}$.

Since a projector on the transformed constraint space is not formally available in the context of compression, it has been suggested by Brie to perform a compression that preserves the constraint space, inspired by Non-negative Matrix Factorization. This compression can be formalized for instance as a constrained approximate Tucker decomposition:

$$\begin{cases} \boldsymbol{\mathcal{T}} = (\boldsymbol{U}\otimes\boldsymbol{V}\otimes\boldsymbol{W})\,\boldsymbol{\mathcal{G}} + \boldsymbol{\mathcal{E}} \\ \boldsymbol{\mathcal{G}} \geq 0,\ \boldsymbol{U} \geq 0,\ \boldsymbol{V} \geq 0,\ \boldsymbol{W} \geq 0 \end{cases} \tag{3.22}$$

where $\boldsymbol{\mathcal{G}}$ is smaller than $\boldsymbol{\mathcal{T}}$. Then a non-negative CPD can be run on non-negative $\boldsymbol{\mathcal{G}}$ and applying operator $\boldsymbol{U}\otimes\boldsymbol{V}\otimes\boldsymbol{W}$ will not remove non-negativity during the decompression step. However it is possible that such a heavily constrained model has no solution, and conditions ensuring that this model is well-posed are not known.

On the other hand, using HOSVD to compress the tensor in an unconstrained fashion yields a well-posed optimization problem. This is however not trivial. Indeed, non-negativity constraints make approximate CPD well-posed since the best low rank approximation exists in a closed space, but for the compressed non-negative model

$$\begin{cases} \boldsymbol{\mathcal{G}} = (\boldsymbol{A}_c\otimes\boldsymbol{B}_c\otimes\boldsymbol{C}_c)\,\boldsymbol{\mathcal{I}}_R + \boldsymbol{\mathcal{E}}_c \\ \boldsymbol{W}\boldsymbol{C}_c \geq 0 \end{cases} \tag{3.23}$$

there may be no solution. That is, the set $\{\boldsymbol{C}_c \in \mathbb{R}^{R_3\times R} \mid \boldsymbol{W}\boldsymbol{C}_c = \boldsymbol{C}\}$ may be empty. In appendix C, it is proved that this is not the case, *i.e.* the HOSVD compression can be used in the context of non-negativity constraints, and the best low rank approximate of the core tensor $\boldsymbol{\mathcal{G}}$ exists while satisfying the transformed constraints. Credits for the proof go to Konstantin Usevich.

A discussion on existing algorithms for constrained compressed CPD

For the sake of argument, two algorithms specific to the non-negative compressed CPD are presented. In the following it is shown how these algorithms are adapted in this particular context. Through the mitigated results these algorithms provide, the need for a more general and more efficient decomposition algorithm is stressed out.

A first solution to compute an approximate compressed non-negative decomposition is to cast the optimization problem in terms of sequential Quadratic Programming problems. Indeed, with respect to variable $\boldsymbol{C}_c$, (3.23) implies the following optimization problem:

$$\begin{array}{l} \text{minimize } \|\boldsymbol{\mathcal{G}} - (\boldsymbol{A}_c\otimes\boldsymbol{B}_c\otimes\boldsymbol{C}_c)\,\boldsymbol{\mathcal{I}}_R\|_F^2 \\ \text{w.r.t. } \boldsymbol{C}_c \\ \text{s.t. } \boldsymbol{W}\boldsymbol{C}_c \geq 0 \end{array} . \tag{3.24}$$

[2] A very small value can be used instead of 0 to avoid numerical complications. For instance, in MATLAB2014b™, use "eps".

Since the cost function is quadratic (recall the CPD is linear with respect to each factor), (3.24) is an instance of the Quadratic Programming framework. A usually good algorithm to minimize (3.24) is the active set method, where constraints generate a dual space using the Lagrangian operator, and only active constraints are used to solve the optimization problem at each iteration. The results of applying the *quadprog* toolbox from MATLAB2014b$^{\text{TM}}$ to the non-negative tensor compression problem are shown in Figure 3.2. Performance of the compressed algorithm is compared in both speed an accuracy with the workhorse algorithm ANLS and other algorithms presented below. Active set does succeed at solving the compressed constrained tensor decomposition but at the cost of large computation time. This is expected since the number of constraints is huge with respect to the number of parameters, and quadratic programming methods require multiple iterations at each iteration of the compressed ALS. Also the Quadratic Programming framework does not cover other type of non-linear constraints. A major criticism here is that the MATLAB toolbox was used as a black box, whereas other algorithms have been reprogrammed manually. In particular, the *quadprog* toolbox works only with vectorized data, so that Kronecker products appear in the vectorized problem formulation. This certainly slows down the active-set method considerably. An adaptation of the FNNLS algorithm should instead be implemented.

Another solution is to resort to projected algorithms. If used in coordination with ALS in a similar fashion than ANLS, after finding the best estimate for $\boldsymbol{C}_c^{(1/2)}$ in the compressed domain, the projection problem under non-negativity constraints is the following:

$$\begin{array}{l} \underset{\boldsymbol{X}}{\operatorname{argmin}} \|\boldsymbol{X} - \boldsymbol{C}_c^{(1/2)}\|_F^2 \\ \text{s.t. } \boldsymbol{W}\boldsymbol{X} \in \mathcal{S}_C \end{array} . \tag{3.25}$$

In the case of non-negativity constraint for instance, no exact projector is known. Thus this projection may only be computed through iterative algorithms. A well known solution to (3.25) is the Dykstra algorithm [14], although a previous simpler algorithm by Hildreth [63] also solves (3.25) in an iterative manner. In both methods, constraints are applied sequentially by projecting on halfspaces defined by rows of $\boldsymbol{W}$. For non-negativity constraints, Dykstra's algorithm solves the following optimization problem:

$$\begin{array}{l} \underset{\boldsymbol{X}}{\operatorname{argmin}} \|\boldsymbol{X} - \boldsymbol{C}_c^{(1/2)}\|_F^2 \\ \text{s.t. } \forall i \in [1; M],\ \boldsymbol{w}_i\boldsymbol{X} \geq 0 \end{array} \tag{3.26}$$

where $\boldsymbol{w}_i$ is the *ith* row of $\boldsymbol{W}$. Dykstra's algorithm projects sequentially on each half-space defined by $\boldsymbol{w}_i$ and tries to find the intersection of all the linear constraints using online correction of the projected vectors. The point is that a simple projector on a halfspace is known for any $\boldsymbol{w}$, by opposition to projection on the transformed non-negative orthant:

$$\Pi_{\{\boldsymbol{w}x>0\}}(\boldsymbol{X}) = \begin{cases} \boldsymbol{X} - \frac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{w}\boldsymbol{X}}{\|\boldsymbol{w}\|_2^2} & \text{if } \boldsymbol{w}\boldsymbol{X} < 0 \\ \boldsymbol{X} & \text{if } \boldsymbol{w}\boldsymbol{X} \geq 0 \end{cases} . \tag{3.27}$$

However constraints are only shown to be satisfied asymptotically so that a great number of iterations of sequential projections may have to be used in practice. Since (3.26) is meant to be solved in each iteration of the ALS algorithm, the running time of these subspace projection methods is prohibitive, especially since in the compressed CPD, many constraints are imposed on the compressed factors. Figure 3.2 compares Dykstra's algorithm applied to non-negative compressed CPD with other algorithms studied in this section and show that it is quite slow to project sequentially on each halfspace.

3.2.2 An alternate projection method: PROCO-ALS

Algorithm description and discussion on convergence

A naive approach to solving (3.24) is to use operator (3.21) as a projection operator in the general case, even though it is not *stricto-sensus* a projection operator on the initial constraint space since $\boldsymbol{W}$ is not orthogonal on the right, *i.e.* $\boldsymbol{W}\boldsymbol{W}^{\mathsf{T}} \neq \boldsymbol{I}$. Algorithm 6 obtained by using such a pseudo-projection in the ALS steps is called PROjected-COmpressed ALS (PROCO-ALS) and was introduced in [32].

In PROCO-ALS, optimization is done in the compressed space using the ALS algorithm, but every time a constrained factor is linearly updated, the constraint is applied in the uncompressed

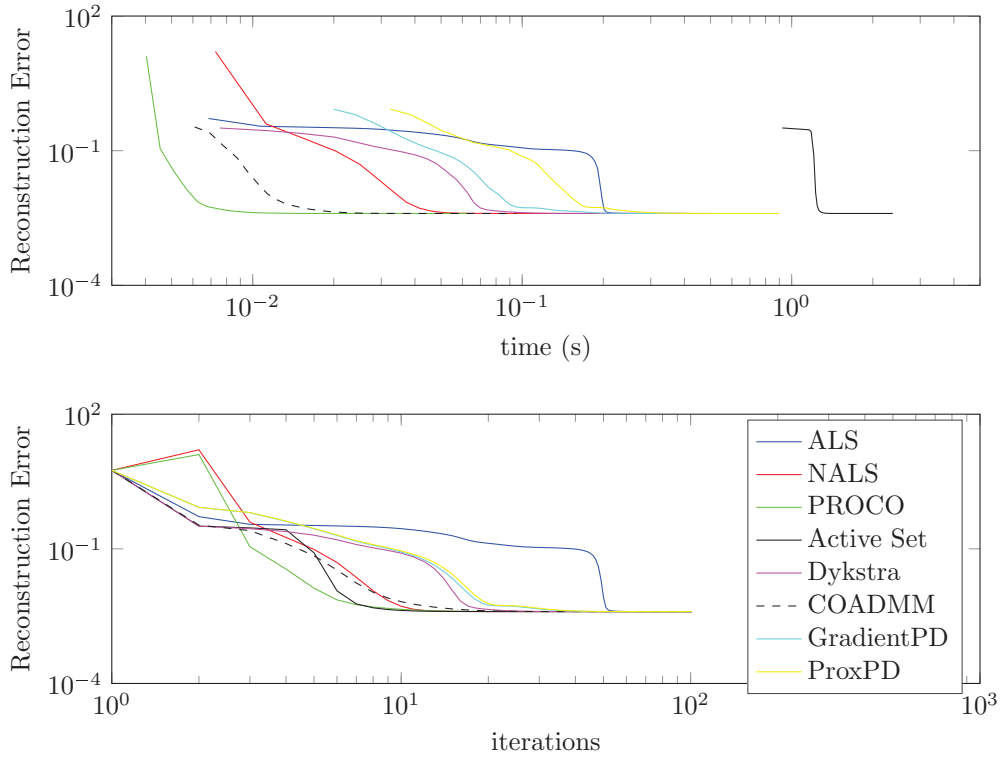


Figure 3.2: A comparative run for all algorithms presented in this chapter. A rank 4 tensor is generated randomly through normalized Gaussian distributed factors of dimensions 100 per 4. SNR is set to 30dB. Tensor is compressed to a $4 \times 4 \times 4$ core tensor through three SVDs. 10 inner loops where used for the Active set algorithm, 2 for Dykstra algorithm and COADMM introduced in section 3.2.4 (with $\rho = 0.01$) and 5 for the Primal-Dual algorithms, with $\rho = 1.95$, $\tau = 0.45$ and $\sigma = 0.01$. Maximal number of iterations is set to 100. Top plot shows the reconstruction error $\|\boldsymbol{\mathcal{T}} - \hat{\boldsymbol{\mathcal{T}}}\|_F^2$. For compressed algorithms, the HOSVD running time is not included.

Algorithm 6 PROCO-ALS

INPUT: array $\boldsymbol{\mathcal{T}}$, compression matrices $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{W}$ and initial factors $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$.
Compress $\boldsymbol{\mathcal{T}}$ into $\boldsymbol{\mathcal{G}}$ by $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{T}} \bullet_1 \boldsymbol{U}^\mathsf{T} \bullet_2 \boldsymbol{V}^\mathsf{T} \bullet_3 \boldsymbol{W}^\mathsf{T}$
Compress initial guesses $\boldsymbol{A}_c = \boldsymbol{U}^\mathsf{T}\boldsymbol{A}$, $\boldsymbol{B}_c = \boldsymbol{V}^\mathsf{T}\boldsymbol{B}$, $\boldsymbol{C}_c = \boldsymbol{W}^\mathsf{T}\boldsymbol{C}$
while convergence criterion is not met **do**

$$\boldsymbol{A}_c = \boldsymbol{\mathcal{G}}_{(1)} \left(\boldsymbol{B}_c \odot \boldsymbol{C}_c\right) \left(\boldsymbol{B}_c^\mathsf{T}\boldsymbol{B}_c \boxdot \boldsymbol{C}_c^\mathsf{T}\boldsymbol{C}_c\right)^{-1}$$
$$\boldsymbol{B}_c = \boldsymbol{\mathcal{G}}_{(2)} \left(\boldsymbol{A}_c \odot \boldsymbol{C}_c\right) \left(\boldsymbol{A}_c^\mathsf{T}\boldsymbol{A}_c \boxdot \boldsymbol{C}_c^\mathsf{T}\boldsymbol{C}_c\right)^{-1}$$
$$\boldsymbol{C}_c^{(1/2)} = \boldsymbol{\mathcal{G}}_{(3)} \left(\boldsymbol{A}_c \odot \boldsymbol{B}_c\right) \left(\boldsymbol{A}_c^\mathsf{T}\boldsymbol{A}_c \boxdot \boldsymbol{B}_c^\mathsf{T}\boldsymbol{B}_c\right)^{-1}$$
$$\boldsymbol{C}_c = \boldsymbol{W}^\mathsf{T} \Pi_{\mathcal{S}_C} \left(\boldsymbol{W}\boldsymbol{C}_c^{(1/2)}\right)$$

end while
OUTPUT: Uncompressed estimated factors $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{A}_c$, $\boldsymbol{B} = \boldsymbol{V}\boldsymbol{B}_c$, $\boldsymbol{C} = \Pi_{\mathcal{S}_C}\left(\boldsymbol{W}\boldsymbol{C}_c\right)$

space by decompressing the factor, projecting it onto the constraint space in the uncompressed domain, and compressing it back. As discussed earlier, this procedure does not ensure that the compressed factor lies in the transformed constraints space. Moreover, because the cost function may not decrease at each step of PROCO-ALS, just like ANLS, there is no guaranty that PROCO-ALS converges to at least a stationary point.

However it is still possible to discuss the convergence of PROCO-ALS. To ensure that a local minimum of the constrained ALS is also a fixed point of PROCO-ALS, a minimizer $\boldsymbol{C}_c$ has to generate $\boldsymbol{X} = \Pi_{\mathcal{S}_C}(\boldsymbol{W}\boldsymbol{C}_c)$ in the image of $\boldsymbol{W}\boldsymbol{W}^\mathsf{T}$. There is no theoretical proof that this is true. However a sufficient condition is that after a certain number of iterations, $\boldsymbol{W}\boldsymbol{C}_c$ is always in $\mathcal{S}_C$, which is verified in practice, see Experiments below. Indeed, this implies that $\Pi_{\boldsymbol{W}(\mathcal{S}_C)} = id$, and PROCO-ALS is simply an ALS running in the compressed domain. Since the convergence of ALS itself is an active research topic, having an algorithm that works in practice but with few theoretical results on convergence is acceptable. If better convergence properties are needed for the application at hand, algorithms presented in subsection 3.2.5 can be used instead.

An intuition on PROCO-ALS is that the quality of the final estimate after decompression depends heavily on the compression quality. If no compression is done, PROCO-ALS is exactly a projected algorithm. If compression is almost without loss, the hope is that an algorithm using the pseudo-projection $\Pi_{\boldsymbol{W}(\mathcal{S}_C)}$ achieves similar performances than a projected algorithm. Further research on how the compression error relates to convergence properties of PROCO-ALS is yet mandatory.

Finally, it can be shown easily that the pseudo-projection step always decreases the estimation error on the factors. This does not prove PROCO-ALS converges if ALS converges since ALS minimizes the reconstruction error, but at least this guarantees the pseudo-projection reduces the error on the factors.

Proposition 4 *In the compression-decompression iteration described above, if $\boldsymbol{C}_c$ denotes the true compressed factor $\boldsymbol{W}^\mathsf{T}\boldsymbol{C}$, it holds that*

$$\left\|\Pi_{\boldsymbol{W}(\mathcal{S}_C)}\left(\hat{\boldsymbol{C}}_c^{(1/2)}\right) - \boldsymbol{C}_c\right\|_F^2 \leqslant \left\|\hat{\boldsymbol{C}}_c^{(1/2)} - \boldsymbol{C}_c\right\|_F^2. \tag{3.28}$$

The proof starts with a simple well-known lemma

Lemma 1 *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two convex closed sets of $\mathbb{R}^N$, with a non empty intersection, $\boldsymbol{x_o}$ be a vector in $\mathcal{S}_1 \cap \mathcal{S}_2$, and p_i denote the projector onto $\mathcal{S}_i$. Then we have, $\forall \boldsymbol{x} \in \mathbb{R}^N$:*

$$\|p_2 \circ p_1\boldsymbol{x} - \boldsymbol{x}_o\| \leqslant \|\boldsymbol{x} - \boldsymbol{x}_o\|,$$

where $\|\cdot\|$ denotes the Euclidean norm.

The proof of lemma 1 is straightforward: $\|p_2 \circ p_1\boldsymbol{x} - \boldsymbol{x}_o\| = \|p_2 \circ p_1\boldsymbol{x} - p_2 \circ p_1\boldsymbol{x}_o\|$ because $\boldsymbol{x}_o \in \mathcal{S}_1 \cap \mathcal{S}_2$, and since p_i are contracting $\|p_2 \circ p_1\boldsymbol{x} - p_2 \circ p_1\boldsymbol{x}_o\| \leqslant \|\boldsymbol{x} - \boldsymbol{x}_o\|$.

Apply the lemma with $\mathcal{S}_1 = \mathcal{S}_C$ so that $p_1 = \Pi_{\mathcal{S}_C}$, and with $\mathcal{S}_2 = \mathrm{Span}\{\boldsymbol{W}\}$ being the subspace spanned by the R_1 columns of matrix $\boldsymbol{W}$. In that case, $p_2 = \boldsymbol{W}\boldsymbol{W}^T$. Then for any non-negative matrix $\boldsymbol{X}_o$ of $\mathcal{S}_1 \cap \mathcal{S}_2$, and for any matrix $\boldsymbol{X}$ of $\mathbb{R}^{K\times R}$, we have:

$$\|\boldsymbol{W}\boldsymbol{W}^T\Pi_{\mathcal{S}_C}\boldsymbol{X} - \boldsymbol{X}_o\|_F \leqslant \|\boldsymbol{X} - \boldsymbol{X}_o\|_F.$$

Yet, as any element of $\mathcal{S}_2$, $\boldsymbol{X}_o$ can be written as $\boldsymbol{X}_o = \boldsymbol{W}\boldsymbol{M}_o$, where $\boldsymbol{M}_o$ is a matrix of size $R_1 \times R$. Now apply this result to a general element of $\mathcal{S}_2$, $\boldsymbol{X} = \boldsymbol{W}\boldsymbol{M}_c$. We get the inequality:

$$\|\boldsymbol{W}^T\Pi_{\mathcal{S}_C}\boldsymbol{W}\boldsymbol{M}_c - \boldsymbol{M}_o\|_F \leqslant \|\boldsymbol{M}_c - \boldsymbol{M}_o\|_F \tag{3.29}$$

which holds true because $\boldsymbol{W}$ is an isometry, that is, because $\|\boldsymbol{W}\boldsymbol{y}\| = \|\boldsymbol{y}\|$. ▫

This property is verified only for orthonormal compression matrix $\boldsymbol{W}$. In other words, using PROCO-ALS directly as presented above is possible for other compression methods than HOSVD, but orthogonality of the chosen compression matrix is still necessary.

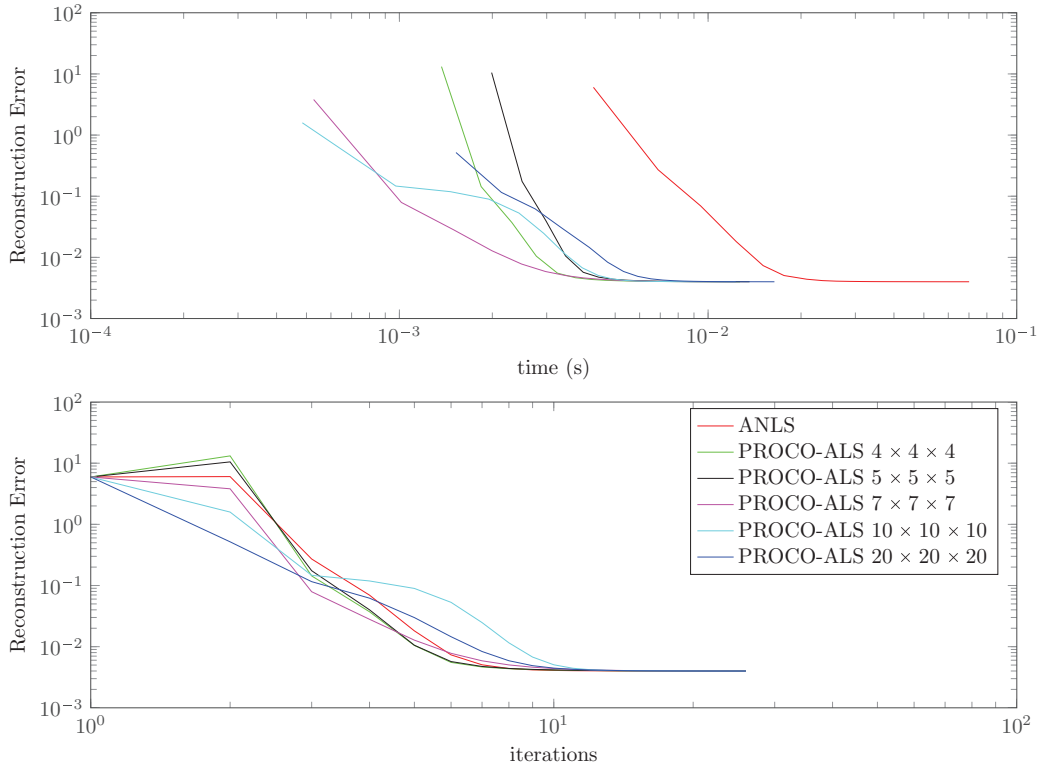


Figure 3.3: Best run out of 3 runs. Array is $100 \times 100 \times 100$ of rank 4, SNR is set to 30dB. Random positive initialization is achieved by absolute values of centered Gaussian, and noise follows a white centered Gaussian distribution. Compression time not included.

Experiments with PROCO-ALS

In order to check the performances of PROCO-ALS, the algorithm is tested on both simulated data and fluorescence multiway data. Simulated data follow exactly the non-negatively constrained CPD to check the good behavior of PROCO-ALS with respect to ANLS. The fluorescence data set is described in section 2.1.2 and serves the purpose to compare PROCO-ALS with ANLS, an algorithm known to be extremely efficient at precisely unmixing interpretable factors with reasonable computation time.

Figures 3.3 and 3.4 show that PROCO-ALS performs well for data with high SNR, both in terms of speed and final reconstruction error. These two plots show in particular that a maximal compression is not necessary to speed-up the tensor decomposition. On the contrary, there is no significant difference between PROCO-ALS runs with small variations of the compressed sizes, so that the amount of compression to be used in practice is a parameter that does not need to be tuned precisely, and can be set a little higher than R. Table 3.1 shows the reconstruction error on the factors for different SNR. The loss of accuracy on the recovered factors due to compression is extremely small here. This is expected since the simulated data follows exactly an approximate low rank CPD model with Gaussian noise and high SNR, which is the scenario for which PROCO-ALS is built. Should the noise distribution be non-Gaussian, results could vary drastically since compression could be significantly lossy.

In order to prove the efficiency of PROCO-ALS as a decomposition algorithm, it is mandatory to study its performance for decomposing constrained real tensor data. For this experiment, fluorescence data described in section 2.1.2 are used. Initial guesses for factors are coefficient-wise absolute values of randomly sampled factors, drawn according to a standard Gaussian distribution[3].

First the impact of compression on the final reconstruction error is again studied, along with the impact of compression on the shape of estimated factors. Since the rank of the fluorescence data is known to be 4, the tensor can in theory be compressed up to at least $4 \times 4 \times 4$. Moreover, two of the dimensions are quite small (23 and 28) so that it makes sense to run compression only

[3] A smarter way to initialize would be to start directly in the intersection of column space of matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ with the constraint spaces, but this proved unnecessary in our experiments.

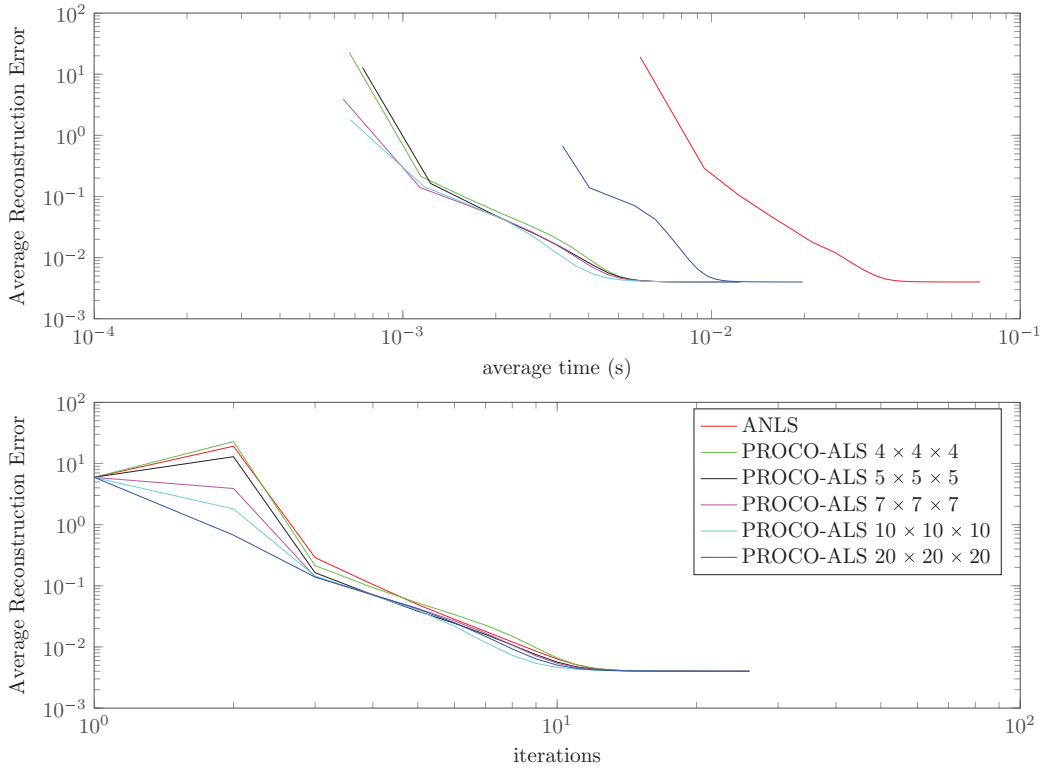
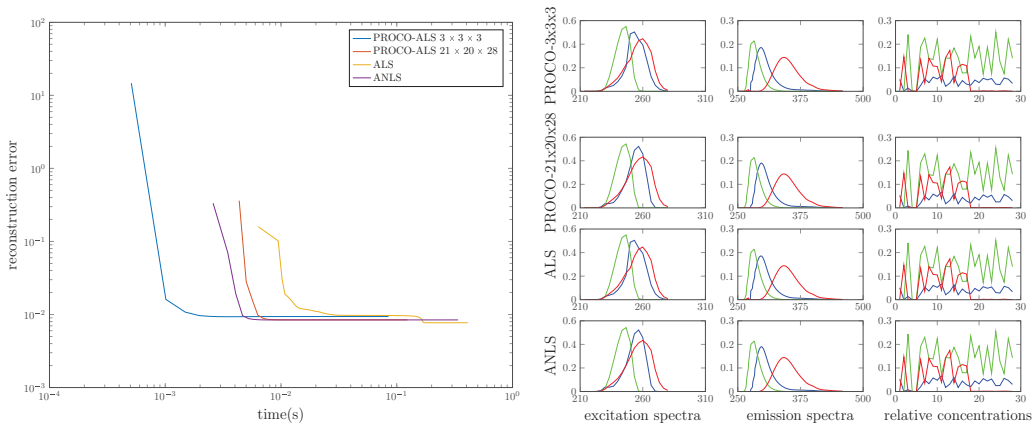


Figure 3.4: Average of the best CPD out of 3 runs for 30 arrays (Figure 3.3 averaged 30 times). Arrays are $100 \times 100 \times 100$ of rank 4, SNR is set to 30dB. Random positive initialization is achieved by absolute values of centered Gaussian, and noise follows a white centered Gaussian distribution. Compression time not included.

		SNR= 30dB			SNR= 10dB		
Algorithm	Sizes	$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{C}$	$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{C}$
PROCO-ALS	$4 \times 4 \times 4$	1.5	1.4	5.2	2.6	2.5	3.3
	$5 \times 5 \times 5$	1.3	1.2	4.6	2.6	2.5	3.3
	$7 \times 7 \times 7$	1.6	1.4	5.5	2.5	2.4	3.0
	$10 \times 10 \times 10$	1.6	1.5	5.9	2.5	2.4	3.1
	$20 \times 20 \times 20$	1.4	1.4	5.1	2.5	2.4	2.9
ANLS	$100 \times 100 \times 100$	1.2	1.1	3.5	2.5	2.4	3.0
		$\times 10^{-5}$			$\times 10^{-4}$		

Table 3.1: Mean square error on reconstructed factors for data simulated as in Figure 3.4.



(a) (b)

Figure 3.5: Best run for 10 different initializations on the data set presented in section 2.1.2 for rank set to 3. Fluorescence spectroscopy is supposed to recover 3 amino-acids spectra : Val-Tyr-Val, Trp-Gly, Phe. (a) Reconstruction error $\|\boldsymbol{\mathcal{T}} - \hat{\boldsymbol{\mathcal{T}}}\|_F^2$ for PROCO-ALS with small and medium sizes, ALS and ANLS. HOSVD computation time is not included. (b) Factors obtained out of the best run for each tested algorithm. Each colored plot stands for a column of the recovered factors.

on the second mode of dimension 251 for comparison. Compressed dimensions are then set to 20. Results are reported in Figure 3.5. Second, computation time for each algorithms is compared also in Figure 3.5, with similar results than with synthetic data. Although the maximally compressed PROCO-ALS is the fastest algorithm by far, it provides different results than the ANLS for one of the excitation spectra. The difference is small but can be significant depending on the application. On the other hand, the less compressed PROCO-ALS outputs the same factors as ANLS but at the cost of computation time. ALS performs also quite well since this data set follows rather precisely an approximate low rank CP decomposition model[4]. As a conclusion, there can be a trade-off between more compression, which improves computation speed, and less compression which may improve interpretability of the results.

Finally, let us see how well PROCO-ALS satisfies the linear constraints and the non-negativity constraints on both simulated and real data. Actually, since the projection on the non-negative orthant is always the last step of the algorithm, the question to be asked is how close from the column span of $\boldsymbol{W}$ does $\hat{\boldsymbol{C}} = [\boldsymbol{W}\boldsymbol{C}_c]^+$ lie. A way to measure this proximity is by comparing the norm of $\hat{\boldsymbol{C}}$ with the norm of its projection on the column span of $\boldsymbol{W}$. The following indicator is used:

$$\eta(\boldsymbol{C}) = \frac{\|(\boldsymbol{W}^\mathsf{T}\boldsymbol{W})^{-1}\boldsymbol{W}^\mathsf{T}\boldsymbol{C}\|_F^2}{\|\boldsymbol{C}\|_F^2}. \tag{3.30}$$

Since any projector is a contraction, η is always smaller than 1. Moreover, the closer η is from 1, the closer $\boldsymbol{C}$ is from the column span of $\boldsymbol{W}$. For both simulated and fluorescence data, $\eta(\boldsymbol{C})$ becomes close to 1 up to machine precision after very few iterations. This proves that PROCO-ALS is doing its job by finding a solution in the intersection of the column space of $\boldsymbol{W}$ and the non-negative orthant.

3.2.3 Proximal primal-dual algorithms

The main issue with PROCO-ALS is that there is no guaranty of convergence of the iterates, let alone convergence to a local minimum of the cost function 3.25. The algorithm itself is based on the approximation of a projection operator but how well this approximation performs theoretically is not clear. Thus there is a need for further algorithms to compute constrained compressed CPD that offer some theoretical guaranty. This section studies the use of proximal algorithms to solve the compressed constrained CPD.

[4]ANLS starts faster than ALS since for a few first iterations there are many zeros in the factors in ANLS, so that matrix multiplications are computed more efficiently and the linear systems can be solved faster. Applying unconstrained compression to ALS would however make it comparable to PROCO-ALS in terms of speed.

Proximal algorithms have gained in popularity in the recent years through their efficient handling of large data while ensuring convergence of the iterates to the global minimum of a convex cost function [13]. There are a multitude of proximal methods to solve convex and non-convex optimization problems, among which the proximal gradient, the accelerated proximal gradient FISTA and the dual counterpart of the Douglas Rachford algorithm *i.e.* the renowned ADMM, have been investigated in the constrained tensor decomposition literature [80, 126].

Proximal algorithms solve the following kind of problem:

$$\begin{aligned} &\underset{\boldsymbol{x}}{\operatorname{argmin}} f(\boldsymbol{x}) + g(\boldsymbol{x}) \\ &\text{s.t. } \boldsymbol{x} \in \mathcal{S} \end{aligned} \tag{3.31}$$

where f is a differentiable convex function, and the proximal operator of g and a projector on $\mathcal{S}$ can be computed. By computing the proximal operator of $\mathcal{S}$ and applying it in some way, proximal algorithms solve (3.31). The proximal operator of a function is defined as:

$$\operatorname{prox}\{f\}(\boldsymbol{u}) = \underset{\boldsymbol{x}}{\operatorname{argmin}} \; f(\boldsymbol{x}) + \|\boldsymbol{x} - \boldsymbol{u}\|_2^2 \tag{3.32}$$

and the proximal operator of $\mathcal{S}$ is the proximal operator of its characteristic function (equals 0 when $\boldsymbol{x} \in \mathcal{S}$, equals $+\infty$ otherwise). The proximal operator of $\mathcal{S}$ is an orthogonal projector on $\mathcal{S}$ if such projector exists. An example of a simple alternating proximal gradient algorithm (ANLS) is provided above in example 5.

When the data tensor is not compressed, any proximal algorithm can be used to solve the constrained CPD problem. It has been shown numerically [68] that applying a proximal algorithm to the whole approximate constrained decomposition problem is less efficient than incorporating the proximal algorithm inside an alternating loop on each factor. Note that this is most likely due to the non-convex nature of the global optimization problem. Thus proximal methods are applied in an alternating fashion as in Algorithm 7.

Algorithm 7 Customizable proximal-based algorithm for CPD with $\boldsymbol{C} \in \mathcal{S}_C$

INPUT: array $\boldsymbol{\mathcal{T}}$, initial factors $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$, and proximal algorithm $\phi(\boldsymbol{Y}, \boldsymbol{M}, \boldsymbol{X}, \mathcal{S})$ minimizing $\|\boldsymbol{Y} - \boldsymbol{M}\boldsymbol{X}\|_F^2$ under constraints $\boldsymbol{X} \in \mathcal{S}$.
while convergence criterion is not met **do**

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{\mathcal{T}}_{(1)} \left(\boldsymbol{B} \odot \boldsymbol{C}\right) \left(\boldsymbol{B}^\mathsf{T}\boldsymbol{B} \boxdot \boldsymbol{C}^\mathsf{T}\boldsymbol{C}\right)^{-1} \\ \boldsymbol{B} &= \boldsymbol{\mathcal{T}}_{(2)} \left(\boldsymbol{A} \odot \boldsymbol{C}\right) \left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A} \boxdot \boldsymbol{C}^\mathsf{T}\boldsymbol{C}\right)^{-1} \\ \boldsymbol{C}^\mathsf{T} &= \phi\left(\boldsymbol{\mathcal{T}}_{(3)}^\mathsf{T}, \left(\boldsymbol{A} \odot \boldsymbol{B}\right), \boldsymbol{C}^\mathsf{T}, \mathcal{S}_C\right) \end{aligned}$$

end while
OUTPUT: Estimated factors $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$

If any proximal algorithm is to be applied in the constrained compressed CPD setting, it is necessary to compute the proximal operator of $\boldsymbol{W}^\mathsf{T}(\mathcal{S}_C)$ since the constraint applies on $\boldsymbol{W}\boldsymbol{C}_c$. As discussed in the previous subsection, $\mathcal{S}_C$ is often built so that a projector can be computed, but knowing a proximal operator for the characteristic function of $\mathcal{S}_C$ does not provide a proximal operator for $\boldsymbol{W}^\mathsf{T}(\mathcal{S}_C)$ unless $\boldsymbol{W}$ is orthogonal on the right [96]. When compressing a tensor, $\boldsymbol{W}$ is only orthogonal on the left.

To tackle this issue, Combettes, Pesquet [36] and Condat [39] proposed primal-dual proximal algorithms that do not require explicit computation of the proximal operator of the transformed constrained set. In particular, algorithm (1) from Condat is simple to implement while providing competitive results with respect to the complex algorithm from Combettes. In (3.31), either f or g can be chosen to be the decomposition fitting term, while the constraint is $\boldsymbol{W}\boldsymbol{C}_c \in \mathcal{S}_C$. This gives birth to two algorithms depending on whether the gradient of the fitting term is used or not, which we call respectively Gradient Primal-Dual ALS and Proximal Primal-Dual ALS. Gradient and Proximal Primal-Dual ALS are described below. Results are presented along PROCO-ALS in the previous subsection, see Figure 3.2.

These algorithms are however not solving the constrained compressed CPD in a satisfactory manner. First, there are three parameters to tune. While some hints are given in the literature on

Algorithm 8 Gradient Primal-Dual ALS (to be used in Algorithm 7)

INPUT: second member $\boldsymbol{Y}$, mixing matrix $\boldsymbol{M}$, initial primal variable $\boldsymbol{X}^{(0)}$ and constraint set $\boldsymbol{W}^{\mathsf{T}}\mathcal{S}_C$
PARAMETERS: τ, ρ, σ, iteration number N and initial dual variable $\boldsymbol{Z}^{(0)}$
for $i = 0..N-1$ **do**

$$\begin{aligned}
\boldsymbol{X}^{(i+1)} &= \boldsymbol{X}^{(i)} + \tau \boldsymbol{M}^{\mathsf{T}}\left(\boldsymbol{Y} - \boldsymbol{M}\boldsymbol{X}\right) - \tau \boldsymbol{W}^{\mathsf{T}}\boldsymbol{Z}^{(i)} \\
\boldsymbol{Z}^{(i+1)} &= \text{prox}\{\mathcal{S}_c\}_\sigma\left(\boldsymbol{Z}^{(i)} + \sigma \boldsymbol{W}\left(2\boldsymbol{X}^{(i+1)} - \boldsymbol{X}^{(i)}\right)\right) \\
\boldsymbol{X}^{(i+1)} &:= \rho \boldsymbol{X}^{(i+1)} + (1-\rho)\boldsymbol{X}^{(i)} \\
\boldsymbol{Z}^{(i+1)} &:= \rho \boldsymbol{Z}^{(i+1)} + (1-\rho)\boldsymbol{Z}^{(i)}
\end{aligned}$$

end for
OUTPUT: $\boldsymbol{X}^{(N)}$

Algorithm 9 Proximal Primal-Dual ALS (to be used in Algorithm 7)

INPUT: second member $\boldsymbol{Y}$, mixing matrix $\boldsymbol{M}$, initial primal variable $\boldsymbol{X}^{(0)}$ and constraint set $\boldsymbol{W}^{\mathsf{T}}\mathcal{S}_C$
PARAMETERS: τ, ρ, σ, iteration number N and initial dual variable $\boldsymbol{Z}^{(0)}$
for $i = 0..N-1$ **do**

$$\begin{aligned}
\boldsymbol{X}^{(i+1)} &= \text{prox}\{\|\boldsymbol{Y} - \boldsymbol{M}\bullet\|_F^2\}_\tau\left(\boldsymbol{X}^{(i)} - \tau \boldsymbol{W}^{\mathsf{T}}\boldsymbol{Z}^{(i)}\right) \\
\boldsymbol{Z}^{(i+1)} &= \text{prox}\{\mathcal{S}_c\}_\sigma\left(\boldsymbol{Z}^{(i)} + \sigma \boldsymbol{W}\left(2\boldsymbol{X}^{(i+1)} - \boldsymbol{X}^{(i)}\right)\right) \\
\boldsymbol{X}^{(i+1)} &:= \rho \boldsymbol{X}^{(i+1)} + (1-\rho)\boldsymbol{X}^{(i)} \\
\boldsymbol{Z}^{(i+1)} &:= \rho \boldsymbol{Z}^{(i+1)} + (1-\rho)\boldsymbol{Z}^{(i)}
\end{aligned}$$

end for
OUTPUT: $\boldsymbol{X}^{(N)}$

how to chose them, there is no best way and using Primal-Dual ALS requires expertise. Second, the convergence of both Primal-Dual ALS is only ensured at each iteration of ALS, but convergence of the global algorithm is still not clear. Convergence proofs were the motivation for using proximal algorithms in the context of constrained CPD, but still call for further research.

Finally, proximal algorithms here do not beat PROCO-ALS in terms of speed. They however tackle a slightly wider spectrum of problems, including for instance total variation regularization on the factors of a CPD. Moreover, primal-dual algorithm directly tackle multiple constraints on the factors, whereas imposing multiple constraints with PROCO-ALS is not straightforward.

3.2.4 ADMM under compression: COADMM

The application of ADMM to constrained compressed CPD is shortly reviewed and applied only to non-negative CPD, being a good solution for solving (3.19). A trial run is presented in Figure 3.2 and comparisons are further conducted below.

Application of ADMM to constrained tensor decompositions has been proposed in [68, 80]. In [80], one iteration of ADMM is carried out for each update, while in [68] ADMM is iterated until convergence for each block. In general ADMM is used to solve the following minimization problem (see [12] for details):

$$\begin{array}{ll}
\text{minimize} & f(\boldsymbol{X}) + g(\boldsymbol{Z}) \\
\text{w.r.t.} & \boldsymbol{X}, \boldsymbol{Z} \\
\text{subject to} & \boldsymbol{D}\boldsymbol{X} + \boldsymbol{E}\boldsymbol{Z} = \boldsymbol{F}
\end{array} \tag{3.33}$$

where $\boldsymbol{X}$ and $\boldsymbol{Z}$ are real matrix variables, $\boldsymbol{D}$, $\boldsymbol{E}$ and $\boldsymbol{F}$ are given real matrices and f and g are convex functions of the elements of $\boldsymbol{X}$ and $\boldsymbol{Z}$.

Standard ADMM k-th iterate is [12]

$$\begin{aligned}\boldsymbol{X}^{k+1} &= \arg\min_{\boldsymbol{X}} L_\rho\left(\boldsymbol{X}, \boldsymbol{Z}^k, \boldsymbol{\mu}^k\right)\\ \boldsymbol{Z}^{k+1} &= \arg\min_{\boldsymbol{Z}} L_\rho\left(\boldsymbol{X}^{k+1}, \boldsymbol{Z}, \boldsymbol{\mu}^k\right)\\ \boldsymbol{\mu}^{k+1} &= \boldsymbol{\mu}^k + \rho\left(\boldsymbol{D}\boldsymbol{X}^{k+1} + \boldsymbol{E}\boldsymbol{Z}^{k+1} - \boldsymbol{F}\right)\end{aligned} \tag{3.34}$$

where $\boldsymbol{\mu}_k$ is a matrix of dual variables, ρ is a nonnegative scalar and L_ρ is the Lagrangian given by

$$L_\rho = f(\boldsymbol{X}) + g(\boldsymbol{Z}) + \frac{\rho}{2}\|\boldsymbol{D}\boldsymbol{X} + \boldsymbol{E}\boldsymbol{Z} - \boldsymbol{F}\|_F^2 + \operatorname{Tr}\left(\left[\boldsymbol{\mu}^\mathsf{T}\left(\boldsymbol{D}\boldsymbol{X} + \boldsymbol{E}\boldsymbol{Z} - \boldsymbol{F}\right)\right]\right). \tag{3.35}$$

To apply (3.34) to problem (3.26), we set $\boldsymbol{X} = \boldsymbol{C}_c$, $\boldsymbol{Z} = \boldsymbol{C}$, $\boldsymbol{D} = \boldsymbol{W}$, $\boldsymbol{E} = \boldsymbol{I}$, $f(\boldsymbol{C}_c) = \|\boldsymbol{\mathcal{G}}_{(3)} - \boldsymbol{C}_c(\boldsymbol{A}_c \odot \boldsymbol{B}_c)^\mathsf{T}\|_F^2$ and $g(\boldsymbol{C})$ equals zero if all elements of $\boldsymbol{C}$ are nonnegative, otherwise it is equal to $+\infty$, *i.e.* g is the characteristic function of $\mathbb{R}_+^{M\times R}$. The inner k-th iterate of the descent update for block $\boldsymbol{C}_c$ becomes

$$\begin{aligned}\boldsymbol{C}_c^{k+1} &= \left(\boldsymbol{G}_{(3)}\left(\boldsymbol{A}_c^k \odot \boldsymbol{B}_c^k\right) - \frac{1}{2}\left[\boldsymbol{W}^\mathsf{T}\left(\boldsymbol{\mu}^k - \rho\boldsymbol{C}^k\right)\right]\right)\left(\left[\left(\boldsymbol{A}_c^{k\mathsf{T}}\boldsymbol{A}_c^k\right)\boxdot\left(\boldsymbol{B}_c^{k\mathsf{T}}\boldsymbol{B}_c^k\right)\right] + \frac{\rho}{2}\boldsymbol{I}\right)^\dagger\\ \boldsymbol{C}^{k+1} &= \left[\frac{1}{\rho}\boldsymbol{\mu}^k + \boldsymbol{W}\boldsymbol{C}_c^k\right]_+\\ \boldsymbol{\mu}^{k+1} &= \boldsymbol{\mu}^k + \rho\left[\boldsymbol{W}\boldsymbol{C}_c^{k+1} - \boldsymbol{C}^{k+1}\right].\end{aligned} \tag{3.36}$$

The update sequence (3.36) can be used in Algorithm 7 to obtain an ADMM-ALS algorithm to compute the non-negative compressed approximated CPD.

3.2.5 Comparisons with state of the art methods

In order to asses the performance of PROCO-ALS as a fast and efficient tensor decomposition method, it is necessary to further compare it with state of the art methods. The chosen methods for comparisons are FNNLS [18], AOADMM [68], ANLS and Tensorlab v3.0 [120], which also features a speed up compatible with non-negativity constraints based on a structured approximation of the data. The comparisons also feature COmpressed ADMM described above since it provides an alternative to PROCO-ALS with better convergence properties. Only simulated data is used in the following, with one tensor of size $100 \times 100 \times 100$ of rank $R = 5$, with factors drawn from a uniform distribution over $[0, 1]$, and normalized so that all the energy is pulled in the third factor $\boldsymbol{C}$. Additive Gaussian noise is applied on the data. The outputs of the algorithms are also normalized so that all the energy is pulled on the estimates of the third factor. Finally, the cost function values over time are averaged over 100 decompositions of the same tensor with random initializations.

First case: right number of components and high SNR First, let us study the ideal case where the rank of the tensor R is known, and the SNR is set to a high value, namely 30dB. The maximal number of iterations in this scenario is set to 100 (250 for tensorlab). Because the SNR is high, compression should be accurate and therefore the compressed dimensions are set to R on all modes.

The results are presented in Figure 3.7. The error plotted on the left is $\|\boldsymbol{\mathcal{T}} - \widehat{\boldsymbol{\mathcal{T}}}\|_F^2$ with $\widehat{\boldsymbol{\mathcal{T}}}$ the reconstructed uncompressed tensor, while the error on the right is the squared error on the factors $\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|_F^2 + \|\boldsymbol{B} - \widehat{\boldsymbol{B}}\|_F^2 + \|\boldsymbol{C} - \widehat{\boldsymbol{C}}\|_F^2$. Because the true factors are normalized, this error is proportional to the relative error by a factor $3R = 15$ in these simulations. Also, the permutation of columns of estimated factors matching at most the true factors is computed before computing the squared error on the factors.

In the optimal scenario of high SNR and accurate knowledge of R, PROCO-ALS performs as well as other methods in terms of factor error, but is faster than uncompressed algorithms by an order of magnitude. It is also faster than the compressed version of AOADMM. Note the slightly worse results of AOADMM.

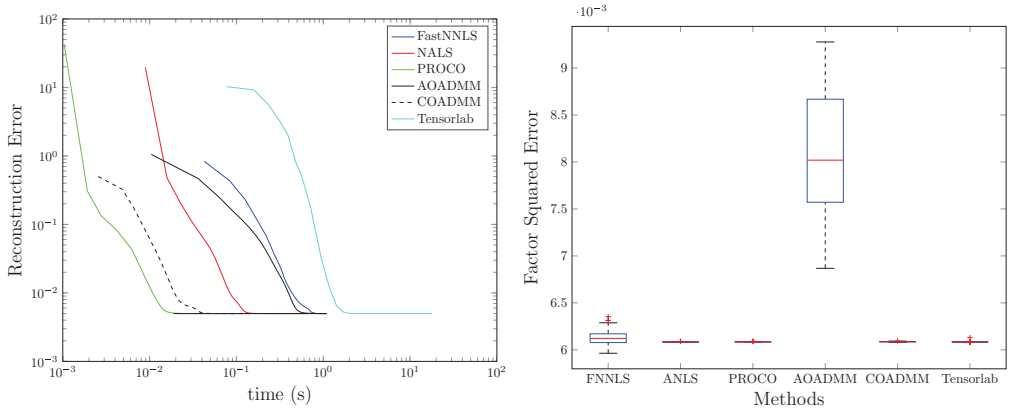


Figure 3.6: Simulation results at high SNR. Left: Reconstruction error plotted against time in seconds, compression time not included. Right: boxplot of the sum of the squared error on each factors.

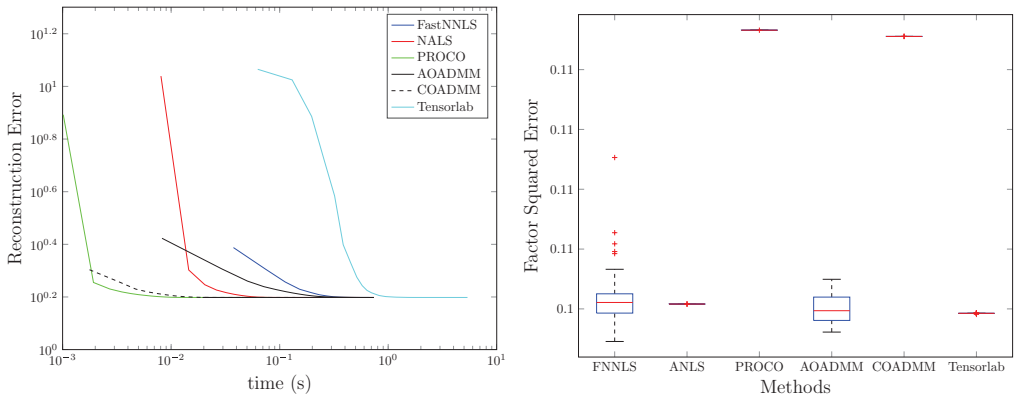


Figure 3.7: Simulation results at low SNR. Left: Reconstruction error plotted against time in seconds, compression time not included. Right: boxplot of the sum of the squared error on each factors.

Second case: right number of components and low SNR In a second experiment, the SNR is set to 5dB, but all the other parameters are left unchanged. Only the compressed dimensions are set a little higher to $R + 3$.

Results are presented in Figure **??**. Compression at low SNR leads to a substantial increase of the error on the factors. On the other hand, the convergence time is not affected and compressed methods still run faster than the uncompressed ones. This means that at low SNR, there is a trade-off between computation time and accuracy on the factors.

Third case: overestimated number of components and high SNR In practice the rank of the data is not known, so that it is likely that the number of components is not well estimated before running decomposition algorithms. Therefore it is mandatory to check the performance of proposed algorithms in this scenario, here with an overestimated number of components $R_{est} = R + 1$. The compressed dimensions are set to $R + 3$, and the maximal number of iterations is set to 200 (still 250 for tensorlab). The SNR is set to 30dB.

Results are presented in Figure 3.8. Five out of six columns of the estimated factors are chosen that best match the true factors. First of all, there is no impact of overestimating the rank on the convergence time. However, FNNLS, ANLS and PROCO-ALS seem to perform better than descent algorithms in this setting, with PROCO-ALS performing surprisingly well. Compressed ADMM outputs unstable results.

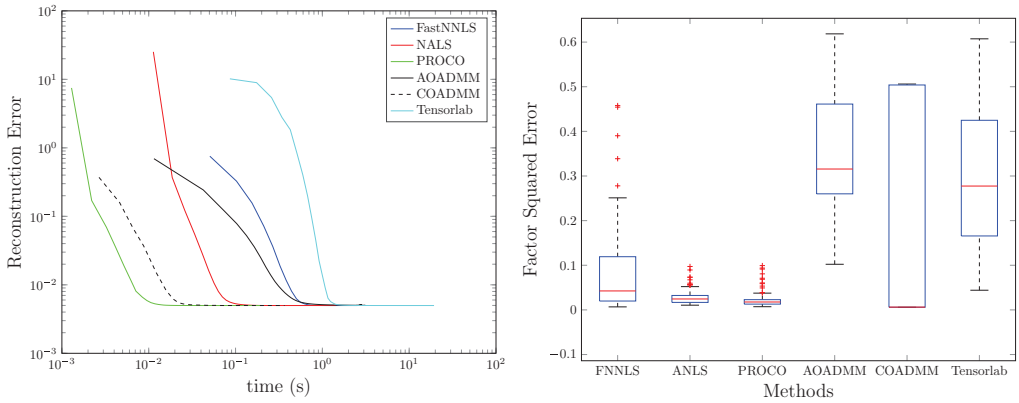


Figure 3.8: Simulation results at high SNR but overestimated number of components, compression time not included. Left: Reconstruction error plotted against time in seconds. Right: boxplot of the sum of the squared error on each factors.

3.3 Dictionary-based CPD

Constrained CPD is concerned with adding prior knowledge on the factors in the tensor decomposition model. Until now in this manuscript only some global knowledge about the factors has been used, like non-negativity, sparsity, smoothness. But it may happen that more precise information is available on the factors. A trivial example is when one of the factor matrices is known. Then it simply needs to be removed from the set of unknown variables in the decomposition model, thus reducing the number of unknown parameters. However what if columns of this factor matrix are only known among a collection of vectors ? That is, a set of available candidates for this factor is known, but which among these candidates are columns of this factor is not known. Another way to understand this problem is to consider that a dictionary of components of one mode is available, therefore finding the factor on this mode amounts to selecting atoms in this dictionary. This last section starts with a formal introduction to the dictionary-based CPD, then it presents algorithms to solve this specific constrained decomposition. An example of application on the Snow data introduced in section 2.2.4 shows a great improvement in interpretability of the results with respect to the fully blind model.

Note that an alternative to using discrete dictionaries is to reparameterize the factors so that they belong to a certain class of functions. Substantial work on these "continuous" dictionaries can be found in [51], with discussion on identifiability and efficient algorithms.

3.3.1 Problem statement

Introduction and formal description

It is common in applied tensor decomposition to have at disposal a benchmark of known shapes for factors. For instance, should the data to be decomposed be a drug test of a professional cyclist, it is likely that doctors will look for spectral signatures of a set of well-known steroids. Dictionary-based CPD allows to incorporate these spectra in the CPD to have proper estimation of the relative concentrations of these components in the mixture.

Another direct application is hyperspectral imaging. In hyperspectral imaging, blind linear unmixing of spectra and abundances is one of the hot topics and gives birth to many challenges. Indeed it is thrilling to estimate both the spectral signatures and the abundance map from a single image provided the number of different materials in the image is small. But even though the blind approach is appealing, it is often useless if a library of spectral signatures is not additionally provided to identify the recovered spectra, since it does not automatically label the recovered components in the spectral factor.

Now if a spectral library (*i.e.* dictionary of spectral signatures) is available, why not use it directly inside the unmixing algorithm instead of resorting to blind unmixing ? There is a certain advantage not to use the dictionary since there may be a few materials in the actual scene that are not reported in the library. However, using dictionaries in the unmixing process should increase the interpretation and accuracy of the unmixing of hyperspectral images since *a priori* information

is added to the model. Below is discussed how to make use of the knowledge of an over-complete dictionary into the CPD.

Let us start with the definition of the dictionary-based CPD.

Definition 17 *The dictionary-based CPD of a tensor $\boldsymbol{\mathcal{T}}$ is the constrained CP decomposition defined as follows:*

$$\begin{cases} \boldsymbol{\mathcal{T}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\, \boldsymbol{\Sigma}_R + \boldsymbol{\mathcal{E}} \\ \boldsymbol{C} = \boldsymbol{D}\boldsymbol{S} \\ \ell_0\left(\boldsymbol{S}\right) = 1 \end{cases} \tag{3.37}$$

where $\boldsymbol{D} \in \mathbb{R}^{M \times D}$ is a known over-complete dictionary, i.e. $D \gg M$, and $\boldsymbol{S} \in \{0,1\}^{D \times R}$ is the score matrix. The ℓ_0 constraint on $\boldsymbol{S}$ is applied columnwise so that each column of $\boldsymbol{S}$ has exactly one non-zero coefficient. The same column (i.e. atom) in $\boldsymbol{D}$ may be selected multiple times in $\boldsymbol{C}$.

This model was first proposed by Sahnoun *et al.* [100] but in the specific context of harmonic retrieval and without the possibility to select multiple times the same atom. A similar idea based on the clustering of columns of $\boldsymbol{C}$ is developed in [123].

In dictionary-based CPD, only one non-zero coefficient is tolerated in each column of $\boldsymbol{S}$. This means that each component in $\boldsymbol{C}$ is exactly a column of the given dictionary $\boldsymbol{D}$. While this requirement may be restrictive, it makes a lot of sense in hyperspectral applications where the role of mixing the components spectra is given to the abundances. However variants of this model can be easily obtained by modifying the constraint on the scores $\boldsymbol{S}$ if multiple atoms are to be associated with each component. Sparsity constraints are yet mandatory, since the unconstrained problem has infinitely many solutions. Indeed, $\boldsymbol{D}$ has more rows than columns, so generically it spans the whole definition space of $\boldsymbol{C}$. This means that the sparsity constraint penalizes the set of possible $\boldsymbol{S}$, and imposes to select only the sparsest.

Relation to dictionary learning

The definition of dictionary-based CPD contrasts with what is usually understood as dictionary learning in the context of tensors. Indeed, some authors [6, 23] refer to the sparsity constrained Tucker model as a dictionary leaning problem, where matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ are interpreted as dictionaries, *i.e.* as a collection of elementary atoms constituting the data. Yet neither the Tucker decomposition nor the CPD generates over-complete families of atoms if factors are tall. On the other hand, dictionaries are usually fat matrices and atoms contain redundant information. Thus it seems weird to impose sparsity constraints on the core tensor in the Tucker model to mimick a dictionary learning model if the obtained dictionaries are tall. On the other hand if the sparse core has larger dimensions than the original tensor, then matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ are fat, but it may be very difficult not to over-parameterize this model. Moreover, the identifiability of over-complete dictionaries is questionable if a single tensor is available for dictionary learning. Similarly to classical dictionary learning, having multiple data set may be the key to Tucker-based dictionary learning:

$$\begin{cases} \boldsymbol{\mathcal{T}}_k = (\boldsymbol{U} \otimes \boldsymbol{V} \otimes \boldsymbol{W})\, \boldsymbol{\mathcal{G}}_k \\ \forall k,\ \ell_0\left(\boldsymbol{\mathcal{G}}_k\right) \leqslant \delta_k \\ \forall k,\ \boldsymbol{\mathcal{G}}_k \in \mathbb{R}^{K'} \otimes \mathbb{R}^{L'} \otimes \mathbb{R}^{M'} \text{ and } K' \gg K,\ L' \gg L,\ M' \gg M \end{cases} \tag{3.38}$$

where $\{\boldsymbol{\mathcal{T}}_k\}$ is a collection of $N > 1$ tensors of same size generated through the same transformation matrices $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ respectively in $\mathbb{R}^{K \times K'}$, $\mathbb{R}^{L \times L'}$ and $\mathbb{R}^{M \times M'}$, and δ_k are given integers. The ℓ_0 pseudo-norm is taken coefficient-wise. Whether this joint decomposition is feasible or not has not been explored in the literature to our knowledge in spite of the great interest this may bear in machine learning.

Actually, (3.38) is a dictionary learning method implied by definition 17. The approach seems different at first since the constrained factor is expressed through a dictionary instead of being the dictionary itself. This usage of dictionaries fits the idea that the underlying parameters are picked out of some over-complete family of atoms. However, dictionary-based CPD of only one tensor is not suited for dictionary learning. Indeed suppose in definition 17 that $\boldsymbol{D}$ is not known. Then even knowing the true $\boldsymbol{C}$, it is not possible to recover both $\boldsymbol{D}$ and $\boldsymbol{S}$ of arbitrary sizes even with the sparsity constraint on the scores $\boldsymbol{S}$. What can be achieved is to cluster columns of $\boldsymbol{C}$ into a

few atoms and find an tall dictionary $\boldsymbol{D}$. On the other hand, if multiple tensors are available, then learning an over-complete dictionary may be possible. Indeed the following holds:

$$\begin{aligned} \boldsymbol{\mathcal{T}}_k &= (\boldsymbol{A}_k \otimes \boldsymbol{B}_k \otimes \boldsymbol{D}\boldsymbol{S}_k)\, \boldsymbol{\Sigma}_R \\ &= (\boldsymbol{I} \otimes \boldsymbol{I} \otimes \boldsymbol{D})\, (\boldsymbol{A}_k \otimes \boldsymbol{B}_k \otimes \boldsymbol{S}_k)\, \boldsymbol{\Sigma}_R \\ &= (\boldsymbol{I} \otimes \boldsymbol{I} \otimes \boldsymbol{D})\, \boldsymbol{\mathcal{G}}_k \end{aligned} \tag{3.39}$$

which is equivalent to the dictionary learning problem (3.38) stated above if the sparsity constraints on cores $\boldsymbol{\mathcal{G}}_k$ are well chosen.

In the remainder of this section, the dictionary is supposed to be known. Learning the dictionary has yet to be explored in future research.

3.3.2 Algorithmic approaches

The actual computation of the dictionary-based CPD of a data array is a rather complex subject. The set of admissible solutions with respect to $\boldsymbol{C}$ is a discrete ensemble made of the columns of the dictionary $\boldsymbol{D}$, so that the optimization problem is highly non-convex. In the matrix case, two well-known algorithm for selecting columns from a dictionary are the Orthogonal Matching Pursuit (OMP) [88] and the LASSO algorithm [110]. Each of these methods has been studied extensively so that a wide variety of algorithms now exist to solve the sparse approximation problem.

On the other hand, for dictionary-based CPD, the paradigm is somewhat different. Indeed there are at least two different ways of understanding the dictionary-based CPD computation. First, since without the knowledge of the dictionary and the sparsity constraint, factor $\boldsymbol{C}$ can be estimated by regular unconstrained CPD, one can alternate between estimation of $\boldsymbol{C}$ by unconstrained least squares and estimation of $\boldsymbol{S}$ through sparse approximation using Matching Pursuit for instance. Another way to compute $\boldsymbol{C}$ is by reparametrizing the problem, looking only for $\boldsymbol{S}$. This can be done by a LASSO algorithm, but it is also simply possible to estimate $\boldsymbol{S}$ through a regularized least squares estimation projected on the closest vertex of the ℓ_1 unit ball.

Since there are many possible ways to optimize (3.37), only two simple methods described in algorithms 10 and 11 have been implemented, namely Projected Penalized ALS and Matching Pursuit ALS. Below, the dictionary-based CPD is applied to the unmixing of the Snow Data.

Algorithm 10 Projected Penalized ALS

INPUT: array $\boldsymbol{\mathcal{T}}$, initial factors $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and dictionary $\boldsymbol{D}$.
while convergence criterion is not met **do**
$\boldsymbol{A}$ and $\boldsymbol{B}$ updates:

$$\boldsymbol{A} = \boldsymbol{\mathcal{T}}_{(1)}\, (\boldsymbol{B} \odot \boldsymbol{C}) \left(\boldsymbol{B}^{\mathsf{T}}\boldsymbol{B} \boxdot \boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}\right)^{-1}$$
$$\boldsymbol{B} = \boldsymbol{\mathcal{T}}_{(2)}\, (\boldsymbol{A} \odot \boldsymbol{C}) \left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} \boxdot \boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}\right)^{-1}$$

$\boldsymbol{S}$ first estimate:

$$\boldsymbol{D}^{\mathsf{T}}\boldsymbol{D}\boldsymbol{S} \left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} \boxdot \boldsymbol{B}^{\mathsf{T}}\boldsymbol{B}\right) + \lambda \boldsymbol{S} = \boldsymbol{\mathcal{T}}_{(3)}\, (\boldsymbol{A} \odot \boldsymbol{B})$$

solved as a Sylvester equation.

$\boldsymbol{S}$ projection step:

$$\forall i < R \;\; s_{ij} = \delta_{j,[\max_j s_{ij}]}; \;\; \boldsymbol{C} = \boldsymbol{D}\boldsymbol{S}$$

end while
$\boldsymbol{C} = \boldsymbol{D}\boldsymbol{S}$
OUTPUT: Estimated factors $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and estimated scores $\boldsymbol{S}$.

3.3.3 Experiments on synthethic and Snow data

Let us check the ability of the two proposed algorithms to solve the dictionary-based CP decomposition on both synthetic and hyperspectral data.

Algorithm 11 Matching Pursuit ALS

INPUT: array $\boldsymbol{\mathcal{T}}$, initial factors $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and dictionary $\boldsymbol{D}$.
while convergence criterion is not met **do**
$\boldsymbol{A}$ and $\boldsymbol{B}$ updates, $\boldsymbol{C}$ first estimate:

$$\boldsymbol{A} = \boldsymbol{\mathcal{T}}_{(1)} (\boldsymbol{B} \odot \boldsymbol{C}) \left(\boldsymbol{B}^{\mathsf{T}} \boldsymbol{B} \boxdot \boldsymbol{C}^{\mathsf{T}} \boldsymbol{C} \right)^{-1}$$
$$\boldsymbol{B} = \boldsymbol{\mathcal{T}}_{(2)} (\boldsymbol{A} \odot \boldsymbol{C}) \left(\boldsymbol{A}^{\mathsf{T}} \boldsymbol{A} \boxdot \boldsymbol{C}^{\mathsf{T}} \boldsymbol{C} \right)^{-1}$$
$$\boldsymbol{C} = \boldsymbol{\mathcal{T}}_{(3)} (\boldsymbol{A} \odot \boldsymbol{B}) \left(\boldsymbol{A}^{\mathsf{T}} \boldsymbol{A} \boxdot \boldsymbol{B}^{\mathsf{T}} \boldsymbol{B} \right)^{-1}$$

$\boldsymbol{S}$ estimate:
for $i = 1$ to R **do**
compute scalar products $\frac{\langle \boldsymbol{C}_i | \boldsymbol{D}_j \rangle}{\|\boldsymbol{C}_i\| \|\boldsymbol{D}_j\|}$ for all j.
select closest D_i from C_i and update according column S_i.
end for
$\boldsymbol{C}$ estimate

$$\boldsymbol{C} = \boldsymbol{D}\boldsymbol{S}$$

end while
OUTPUT: Estimated factors $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and estimated scores $\boldsymbol{S}$.

	$K = 30$	$K = 50$
MPALS	99%	97.3%
PPALS	98.9%	98.9%

Table 3.2: Percentage of perfect decomposition for different number of atoms. Percentages for 1000 runs. Lambda is set to 10^{-2} and maximal number of iterations is 200.

A main question to be asked about PPALS and MPALS is whether or not they are able to find the right atoms in the given dictionary. Indeed the optimization problem can be written as follows,

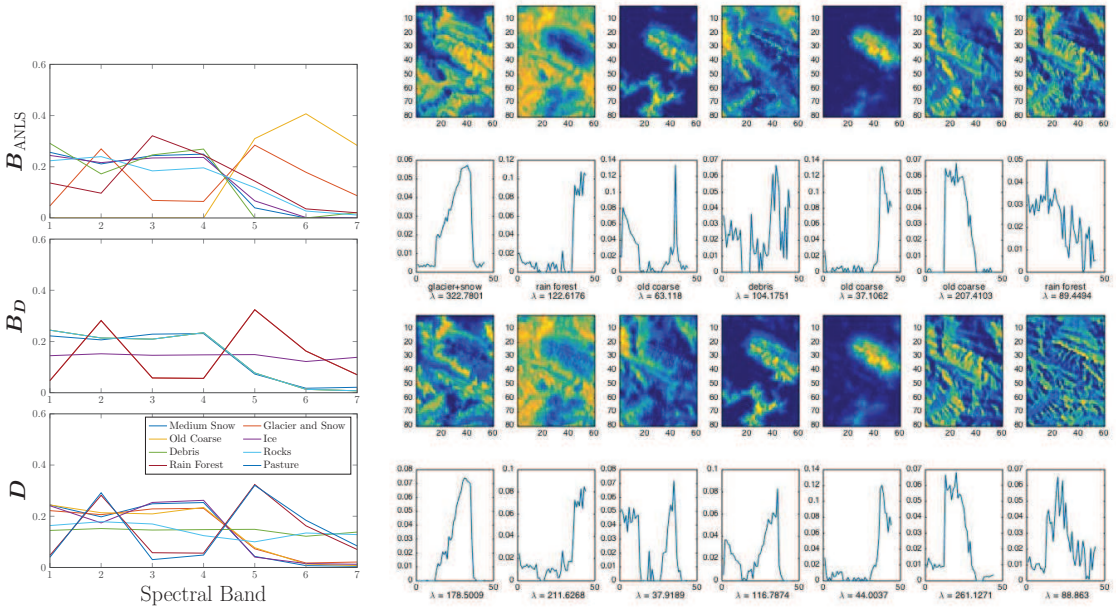
$$\begin{aligned} &\underset{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}}{\operatorname{argmin}} \|\boldsymbol{\mathcal{T}} - (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C}) \boldsymbol{\Sigma}_R\|_F^2 \\ &\text{s.t. } \boldsymbol{C} = \boldsymbol{D}\boldsymbol{S} \text{ and } \ell_0 (\boldsymbol{S}) = 1 \end{aligned} \tag{3.40}$$

and the cost function in (3.40) is not convex since CPD itself is not. Moreover, the fact that columns of $\boldsymbol{C}$ have to be exactly columns of $\boldsymbol{D}$ makes decomposition algorithm very sensitive to local minima if the constraint is satisfied at all steps as in PPALS and MPALS.

To study atom selection performance, a dictionary $\boldsymbol{D} \in \mathbb{R}_+^{K \times D}$ is generated from a coefficient-wise uniform distribution on $[0, 1]$, then normalized columnwise with the ℓ_1 norm. $\boldsymbol{S}$ is chosen to be a diagonal matrix, and other factors are drawn from absolute value of coefficient-wise Gaussian distribution with unit variance. No noise is added, rank is set to 4. The simulated arrays have sizes $30 \times 30 \times 30$. The simulated data were chosen to be non-negative to check performance in a context inherited from hyperspectral images unmixing. Algorithms PPALS and MPALS are modified to include non-negativity constraints through projection on the non-negative orthant. Table 3.2 shows the amount of perfect recovery of all factors depending on the size of the dictionary. PPALS behaves better than MPALS because it makes use of the information provided by the dictionary at every step, while MPALS alternates and is thus further from an optimal algorithm. Notably, in the noiseless context, either the algorithms converge in very few steps, or they do not find the true factors and iterate until the maximal number of iteration is reached. Since it performs better than MPALS, PPALS is used below to compute a dictionary-based CPD on the Snow data set.

Figure 3.9 compares the application of ANLS and dictionary-based CPD computed with PPALS to the Snow data reported in [116]. The parameters of the CP decomposition model used for Figure 3.9 were the following: CP rank R was set to 7, maximum number of iterations was set to 10^3, initial values for factors were drawn with a standard Gaussian distribution coefficient-wise, and λ was set to 10^{-3}. Dictionary $\boldsymbol{D}$ was normalized columnwise. Comparison is done with the same set of parameters and same initial factors.

ANLS provides some interpretable results. By checking the correlation among recovered spectra and the atoms in the dictionary, it can be seen that some identified materials fit with what is



(a) Recovered and Dictionary Spectra

(b) Abundances and Temporal signatures

Figure 3.9: Recovered factors from non-negative CPD versus dictionary-based non-negative CP decomposition.

expected to be found in the scene. The abundance maps are well resolved, and exhibit a snowy mountain surrounded by a plain landscape mainly composed of "green" materials. Even the temporal factor is meaningful, showing an increase in the amount of snow during the winter season. However, some spectra do not match with anything expected to be found in the Alps, *i.e.* some spectra are not columns of the dictionary. This means that a part of the outputs of ANLS is either uninteresting spectra or junk. On the other hand, PPALS cannot find other spectra than what is contained in the dictionary, thus providing clearly interpretable results for all components. Moreover the abundance maps still make sense and seem to allow for a better separation of spatial features. Note that the 'rain forest' and 'pasture' spectra in the dictionary are extremely correlated, which is the reason why rain forest is identified in the middle of the Alps. This atom should have probably been removed from the provided dictionary.

Chapter 4

Understanding subject variability

Summary This chapter deals with the fusion of multiple multiway data sets. The goal is to broadly extend what is usually understood as data fusion with multiway data by exploring the possibility to extract complex interactions between multiple multiway data sets, interactions that make up what we call subject variability. It is structured as follows:

- The subject variability paradigm is exposed, exact coupled CPD is then presented and related to existing models accounting for more complex subject variability.
- A flexible formulation of the data fusion problem is then introduced, which allows for both complex coupling relationships and approximate relationships.
- Compression for coupled data sets is shown to provide good performance, even in a partial coupling context.
- Some open problems related to coupling estimation are exposed.

This chapter is based on some recent publications: the core article on flexible data fusion is joint work with Rodrigo Cabral Farias [21, 22], estimating subject variability in the context of ocular artifact detection in EEG data is joint work with Bertrand Rivet [99], and the compression of coupled tensors is a work realized jointly with Rodrigo [33].

Introduction Modern society is building itself around new sensing technologies. From neuroimaging measuring the functioning of the brain to chemometrics and bioinformatics quantifying body function through spectroscopy and genomics, improved sensing technologies in biomedical engineering have led to an overwhelming mass of data produced every day. Researchers have access to various measurements on large number of subjects along diverse modalities. A major challenge of modern data mining is to exploit these multiple diversities inherent to such massive data sets, and to understand both inter-subject and intra-subject diversity.

When facing large amounts of data on multiple subjects, a natural problem in data processing is to find a common pattern from those various measurements. Usually this common pattern is meant to explain the status of subjects in a comprehensive and concise manner. State-of-the-art data mining methods rely on low dimensional latent representations of high-dimensional data based on feature representations (latent feature models) and characterizations in terms of prototypical individuals (latent class models). However, each subject is unique. When designing treatment for an individual, it is crucial to know how the specific individual will respond to the treatment. Existing low dimensional feature representations, such as principal component analysis (PCA), are in general difficult to interpret since they rely on unjustified assumptions (*i.e.* orthogonality) not optimally designed to leverage multi-subject and multi-modal data. On the other hand, information specifically related to one subject may be contained in the large amount of collected data. People may react differently to a treatment, and understanding this individual variation could change the face of biomedical engineering. In data processing, individual differences with respect to main common patterns are what we call subject variability. Subject variability is a hot topic in biomedical engineering, since understanding subject variability means finding the missing link between Big Data and individualized biomedical signal processing. In the first section, data fusion models that tackle subject variability are presented. The remainder of this chapter exposes contributions on modeling couplings between multiway data sets.

4.1 An introduction to data fusion models accounting for subject variability

4.1.1 Intra/inter-subject variability

To explain what is understood as subject variability in this manuscript, first consider the following simple thought experiment. A baseball player is asked to throw N balls in front of him one after the other, while trying to obtain the same throw. The N trajectories $\{\boldsymbol{s}_i\}_{i\leqslant N}$ are recorded on a 2D grid at a regular scale along the x axis, and in a continuous scale along the y axis so that s_i is a vector containing the height of the ball at each recorded coordinate. Figure 4.1 provides a graphical summary of what data would look like and which underlying processes are going to be estimated.

Having N recordings at disposal, it is very natural to look for a "latent" trajectory $\boldsymbol{s}^*$ that would be hopefully the trajectory the baseball player was picturing when throwing the balls[1]. A whole range of methods exist to estimate $\boldsymbol{s}^*$. Here since the physics of the problem is well known, it is possible to parameterize the trajectory of throws depending on the initial speed and the initial height and to find the parameters for the best fitting $\boldsymbol{s}^*$ in the least squares sense. The distance to be chosen for finding these optimal parameters depends on the distribution of a trajectory with respect to the latent one, for instance least squares are optimal with a Gaussian distribution.

Now the latent trajectory is not the only interesting phenomenon that can be extracted from the data. On the contrary, the player will probably be more interested in knowing how each of his throws differs from the latent trajectory. Indeed understanding how throws vary can help reduce the variance of the throws. In other words, understanding the variability of the throws amounts to recovering the coupling functions f_i linking $\boldsymbol{s}^*$ to each $\boldsymbol{s}_i$:

$$\boldsymbol{s}_i = f_i\left(\boldsymbol{s}^*\right). \tag{4.1}$$

Clearly this problem is not feasible without further constraints, even if $\boldsymbol{s}^*$ is determined by an auxiliary method. However, let us be even more greedy with the model and ask that both f_i and $\boldsymbol{s}^*$ are unknown. At this stage, some hypotheses on f_i are necessary. In particular, finding an invariant in trajectories helps to reduce the research space for the coupling functions f_i.

For instance, assume all trajectories are vertical dilations from each other as in Figure 4.1[2]. Adding the knowledge that all trajectories are parabolic, this invariant can be used to parameterize the coupling functions as follows:

$$f_i(\boldsymbol{x}) = \lambda_i \boldsymbol{x} \tag{4.2}$$

where λ_i are scalars standing for the amount of dilation of each $\boldsymbol{s}_i$ with respect to $\boldsymbol{s}^*$. Since the hypothesis on the coupling functions are strong, one could think that there is a unique set of λ_i and latent trajectory $\boldsymbol{s}^*$ so that $\boldsymbol{s}_i = \lambda_i \boldsymbol{s}^*$. But this is not true.

Actually, when trying to extract information on subject variability without any knowledge on the latent behavior $\boldsymbol{s}^*$, if the set $\mathcal{F}$ of admissible f_i is a group for the composition operator $\circ$ and is not reduced to the identity[3], then $\boldsymbol{s}^*$ can only be found up to a class of equivalence. Indeed, if $\bar{f}$ belongs to $\mathcal{F}$, then supposing $\boldsymbol{s}_i = f_i(\boldsymbol{s}^*)$ for all $i \in [1, N]$,

$$\begin{aligned} f_i(\boldsymbol{s}^*) &= f_i\left(\bar{f}\left(\bar{f}^{-1}\left(\boldsymbol{s}^*\right)\right)\right) \\ &= f_i \circ \bar{f}\left(\bar{f}^{-1}\left(\boldsymbol{s}^*\right)\right) \\ &= f_i^{equiv}(\boldsymbol{s}^{*equiv}) \end{aligned}. \tag{4.3}$$

Thus given a solution $\boldsymbol{s}^*$ to (4.1), the set of solutions for latent trajectories $\boldsymbol{s}^*$ is $\{f(\boldsymbol{s}^*) \mid f \in \mathcal{F}\}$.

For $f_i(\boldsymbol{x}) = \lambda_i \boldsymbol{x}$, the latent trajectory and the coupling functions are thus determined up to a scaling factor. In other words, only the fractions $\frac{\lambda_i}{\lambda_j}$ can be interpreted.

There are two ways to get rid of the indeterminacy of f_i and $\boldsymbol{s}^*$. It is first possible to set $f_1 = id$, so that $\boldsymbol{s}_1 = \boldsymbol{s}^*$, but this is probably not satisfactory if $\boldsymbol{s}^*$ is understood as a typical behavior and should not be chosen arbitrarily. In another spirit, the indeterminacies can be lifted if additional hypotheses can be made on $\boldsymbol{s}^*$. For the baseball experiment, a parametric estimation of $\boldsymbol{s}^*$ in the least squares sense can for instance be used additionally.

[1] The fact that a latent trajectory may not truly exist has to be kept in mind, although this issue will not be discussed here. We consider the existence of a latent trajectory as a modeling hypothesis.

[2] Of course this hypothesis over-simplifies the problem, and a more reasonable one would be that all throws have the same initial kinetic and potential energies. But this last invariant is a little more difficult to exploit while having no pedagogic utility.

[3] $\mathcal{F} = \{id\}$.

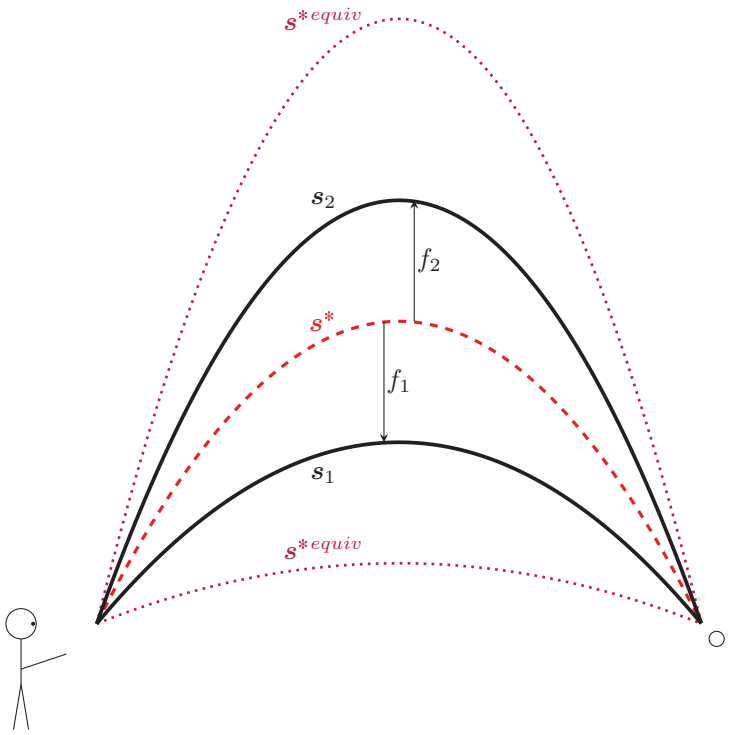


Figure 4.1: A snapshot of 2 throws $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ by a baseball player, in black. An ideal latent trajectory $\boldsymbol{s}^*$ is represented by dashed red line, while two equivalent latent trajectories $\boldsymbol{s}^{*equiv}$ for coupling function $f_1(\boldsymbol{x}) = \lambda_1 \boldsymbol{x}$ and $f_2(\boldsymbol{x}) = \lambda_2 \boldsymbol{x}$ are represented by dotted purple line. Note that there is an infinity of equivalent trajectories, and this includes $\boldsymbol{s}_1$, $\boldsymbol{s}_2$ and $\boldsymbol{s}^*$. In the meantime, the helpful player is contemplating his (thin) muscles.

This whole thought experiment instantiates the study of intra-subject variability, since a single subject (player) emits multiple signals (trajectories) with inherent variability around a latent signal (latent trajectory) that is modeled and estimated. On the other hand, inter-subject variability stands for variations of the recorded signals among multiple subjects. For instance if M players were asked to throw some balls in a given direction, then the latent trajectories $\boldsymbol{s}_m^*$ of each player can be linked with each others if some players throw the balls in a very similar fashion. A popular way to study inter-subject variability is clustering. Since clustering groups the players based on the similarity of their latent trajectories, it requires some a-priori knowledge on a good measurement of similarity between trajectories and a prior estimation of the $\boldsymbol{s}_m^*$. However the goal of this chapter is to go beyond and show that is is possible to learn both a good similarity measure based on invariants in the data and the latent trajectories of each player.

In what follows, the coupling relationship (4.1) is used in the context of multiway data mining. That is, what stands for trajectory is not raw data, but rather estimated factors in the CPD. Moreover, intra-subject variability and inter-subject variability can be studied with exactly the same formalism. Whether the analysis is intra or inter indeed only depends on what is understood as "subject". However the inter-subject and intra-subject variability have been considered in separate models in the literature since these models are often built for specific applications, which in turn require to model either inter or intra-subject variability.

A general framework for studying variability in tensor decompositions goes as follows. Given N tensors $\boldsymbol{\mathcal{T}}_i \in \mathbb{R}^{K_i} \otimes \mathbb{R}^{L_i} \otimes \mathbb{R}^{M_i}$, and assuming factors on the same modes are linked,

$$\forall\ i \in [1, N], \quad \begin{cases} \boldsymbol{\mathcal{T}}_i &= (\boldsymbol{A}_i \otimes \boldsymbol{B}_i \otimes \boldsymbol{C}_i)\boldsymbol{\mathcal{I}}_{R_i} + \boldsymbol{\mathcal{E}}_i \\ \boldsymbol{A}_i &= f_i^{(\boldsymbol{A})}\left(\boldsymbol{A}^*\right), \quad f_i^{(\boldsymbol{A})} \in \mathcal{F}^{(\boldsymbol{A})} \\ \boldsymbol{B}_i &= f_i^{(\boldsymbol{B})}\left(\boldsymbol{B}^*\right), \quad f_i^{(\boldsymbol{B})} \in \mathcal{F}^{(\boldsymbol{B})} \\ \boldsymbol{C}_i &= f_i^{(\boldsymbol{C})}\left(\boldsymbol{C}^*\right), \quad f_i^{(\boldsymbol{C})} \in \mathcal{F}^{(\boldsymbol{C})} \end{cases} \tag{4.4}$$

where $\mathcal{F}^{(\boldsymbol{A})}$, $\mathcal{F}^{(\boldsymbol{B})}$ and $\mathcal{F}^{(\boldsymbol{C})}$ are the set of admissible coupling functions on each mode. Model (4.4) can be derived from a MAP formulation of the estimation of factors in the CPD of multiple

tensors as detailed in section 4.2.1.

Depending on the nature of $\mathcal{F}$, a wide range of models can be specified. For instance, if $\mathcal{F}^{(\boldsymbol{A})}$ is the set of all functions mapping $\mathbb{R}^{K_i \times R_i}$ to $\mathbb{R}^{K \times R}$ and K, R are respectively greater than all K_i, R_i, then factors $\boldsymbol{A}_i$ are unconstrained and the CPDs are unconstrained on the first mode. When studying inter-subject variability, it is natural that only one or two of the modes are related among the tensors, so that the unrelated modes should be unconstrained. On the other hand, if $\mathcal{F}^{(\boldsymbol{A})}$ contains only the identity, then factors $\boldsymbol{A}_i$ are exactly shared among the tensors. This is the topic of the next section.

An even more general setting is obtained if the equalities in (4.4) are replaced to obtain conditional probability distributions on the data tensors and the factors:

$$\forall\ i \in [1, N], \quad \begin{cases} \boldsymbol{\mathcal{T}}_i | \boldsymbol{A}_i, \boldsymbol{B}_i, \boldsymbol{C}_i & \sim \mathcal{D}_i^{(\boldsymbol{\mathcal{T}})} \left((\boldsymbol{A}_i \otimes \boldsymbol{B}_i \otimes \boldsymbol{C}_i) \boldsymbol{\mathcal{I}}_{R_i} \right) \\ \boldsymbol{A}_i | \boldsymbol{A}^* & \sim \mathcal{D}_i^{(\boldsymbol{A})} \left(\boldsymbol{A}^* \right) \\ \boldsymbol{B}_i | \boldsymbol{B}^* & \sim \mathcal{D}_i^{(\boldsymbol{B})} \left(\boldsymbol{B}^* \right) \\ \boldsymbol{C}_i | \boldsymbol{C}^* & \sim \mathcal{D}_i^{(\boldsymbol{C})} \left(\boldsymbol{C}^* \right) \end{cases} \tag{4.5}$$

where $\mathcal{D}_i(\boldsymbol{X})$ are known probability distributions of mean $\boldsymbol{X}$, possibly depending on other parameters than the low rank CPD of each $\boldsymbol{\mathcal{T}}_i$ and the latent factors $\boldsymbol{A}^*$, $\boldsymbol{B}^*$ and $\boldsymbol{C}^*$. This probabilistic model is also derived from the MAP formulation and explored in section 4.2.1.

4.1.2 Exactly shared factors

When several multiway data set are obtained from measuring the same phenomenon, and each data set can be efficiently decomposed as a sum of rank one components, it is likely that similarities exist among the different components. Indeed components in the CPD can be interpreted as latent sources. If different devices measure the same phenomenon, the underlying sources generating the data are the same, but they are observed differently. The most natural idea is probably to suppose these sources are exactly the same in all the data sets, and that they correspond to factors of the CPDs along one mode.

In terms of subject variability modeling, this means that factors on one mode are coupled through $f_i^{(\boldsymbol{C})} = id$, while the other modes are fully dependent on the modality and therefore are not shared:

$$\forall\ i \in [1, N], \quad \begin{cases} \boldsymbol{\mathcal{T}}_i & = (\boldsymbol{A}_i \otimes \boldsymbol{B}_i \otimes \boldsymbol{C}_i) \boldsymbol{\mathcal{I}}_R + \boldsymbol{\mathcal{E}}_i \\ \boldsymbol{C}_i & = \boldsymbol{C}^* \end{cases} . \tag{4.6}$$

This model is coined as 'exact coupled CPD' in the remainder of the manuscript.

Exact coupled CPD has been studied for the first time by Harshman [62], but a recent study of modeling and algorithmic aspects of (4.6) is due to Acar *et. al.* [1, 4]. The identifiability of all parameters in model (4.6) has also been studied recently, without noise $\boldsymbol{\mathcal{E}}_i$ on all tensors [104]. Little is known of the identifiability in the approximate case. Yet it can be easily understood that exact coupling reduces the number of parameters, so that if each independent tensor decomposition problem is well-posed, then the exact coupled model is also well posed. Because the conditions to have identifiable parameters for the CPD are extremely mild in practice, identifiability of unshared and shared factors is not an issue for practical applications. An interesting feature of exact coupled CPD is that, if a second order tensor and a third order tensor are coupled using model (4.6), then the CPD of the second order tensor can be unique (except under scaling and permutation ambiguities among the factors) if the CPD of the third order tensor is unique. Again in practice, three way arrays are most often decomposable into a sum of rank one components, so that exact coupled CPD may restores identifiability of the CPD of the coupled two way array [104].

Moreover there might not be a minimum to the following cost function derived from the maximum likelihood estimator (if noise tensors $\boldsymbol{\mathcal{E}}_i$ are i.i.d. Gaussian[4]):

$$\underset{\boldsymbol{A}_i, \boldsymbol{B}_i, \boldsymbol{C}}{\operatorname{argmin}} \sum_{i=1}^{N} \| \boldsymbol{\mathcal{T}}_i - (\boldsymbol{A}_i \otimes \boldsymbol{B}_i \otimes \boldsymbol{C}) \boldsymbol{\mathcal{I}}_R \|_F^2 \tag{4.7}$$

[4]For the sake of simplicity in this first section of the chapter, noise levels are assumed to be equal for all tensors. Since there is no reason to have i.i.d. observation noises in practice, noise levels are later accounted for. Algorithm 12 can be adapted easily to account for different noise levels, and a noisy version of Algorithm 13 is provided in section 4.4.1.

which in practice is very similar to the best low-rank approximation problem that occurs with the CPD, see section 1.4.1.

One advantage of exact coupled CPD with respect to other coupled CPD models accounting for subject variability is that there is no coupling functions f_i to estimate, which also means that the latent factor $\boldsymbol{C}^*$ is well-defined. Recall that in the general case, $\boldsymbol{C}^*$ is only defined inside a class of equivalence.

But the true advantage of exact coupled CPD, and of any model accounting for subject variability, is that the estimation error on $\boldsymbol{C}$ is reduced with respect to running N independent CPD since coupling reduces the number of parameters and the search space, and inputs prior knowledge in the model. The argument here is very similar to what is discussed in the constrained CPD section 3.1.1, which is fair since all models introduced in this section are constrained decompositions of more than one tensor.

Finally, solving the non-convex optimization problem (4.6) can be done by an alternating least squares algorithm in the same spirit as solving the approximate CPD of multiway data. It is referenced in [3] that for this particular optimization problem, a gradient descent algorithm works better in the case of overfactoring. But since this is supported by no theoretical results, and if overfactoring is not an issue in the problem at hand, the proposed ALS algorithm 12 is satisfactory. Exact coupled CPD is used below in comparison with a flexible coupling model to decompose fluorescence and nuclear magnetic resonance data of the same chemical mixtures.

Algorithm 12 ALS algorithm for solving the exact coupled CPD problem

INPUT: Multiway arrays $\boldsymbol{\mathcal{T}}_i$, initials factors $\boldsymbol{A}_i$, $\boldsymbol{B}_i$ and $\boldsymbol{C}$.

while convergence criterion is not met **do**

for i from 1 to N **do**

$$\boldsymbol{A}_i = \boldsymbol{T}_{i,(1)} \left(\boldsymbol{B}_i \odot \boldsymbol{C}\right) \left(\boldsymbol{B}_i^\mathsf{T}\boldsymbol{B}_i \boxdot \boldsymbol{C}^\mathsf{T}\boldsymbol{C}\right)^{-1}$$

$$\boldsymbol{B}_i = \boldsymbol{T}_{i,(2)} \left(\boldsymbol{A}_i \odot \boldsymbol{C}\right) \left(\boldsymbol{A}_i^\mathsf{T}\boldsymbol{A}_i \boxdot \boldsymbol{C}^\mathsf{T}\boldsymbol{C}\right)^{-1}$$

end for

$$\boldsymbol{C} = \left(\sum_{i=1}^{N} \boldsymbol{T}_{i,(3)} \left(\boldsymbol{A}_i \odot \boldsymbol{B}_i\right)\right) \left(\sum_{i=1}^{N} \boldsymbol{A}_i^\mathsf{T}\boldsymbol{A}_i \boxdot \boldsymbol{B}_i^\mathsf{T}\boldsymbol{B}_i\right)^{-1}$$

end while

OUTPUT: estimated factors $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{C}$.

Before discussing other more complex coupling models, a last subtle difficulty has to be pointed out. If multiway arrays are modeled using exact coupled CPD (4.6), then why was it necessary to cancel out variables $\boldsymbol{C}_i$ in optimization problem (4.7) by substituting them with $\boldsymbol{C}^*$? Indeed it is possible to design an alternating algorithm that extracts $\boldsymbol{C}_i$ and estimates $\boldsymbol{C}^*$ at every iteration.

Here it can be argued that a two-step algorithm is less efficient to minimize the cost function at every iteration than a one-step algorithm like Algorithm 12. But on the other hand estimating $\boldsymbol{C}^*$ in Algorithm 12 is costly, and a full alternating algorithm might provide faster convergence. For completeness, a full alternating algorithm is explicited below in Algorithm 13. It is not clear which algorithm should be preferred over the other.

But what is even harder to determine is whether the latent variables should be estimated or not in general. In the exact coupling CPD this questions the need for estimating $\boldsymbol{C}_i$ and $\boldsymbol{C}^*$ independently. In all models accounting for subject variability introduced in this chapter, whether or not to include a latent factor in the model is actually a difficult question. See section 4.4.1 for a short discussion on this point.

4.1.3 PARAFAC2

In 1972, after having proposed an exact coupled CPD for mining linguistic data [62], Harshman gave birth to yet another coupling model called PARAFAC2 [60]. It is based on the assumption that intra-subject variability is invariant under left orthonormal transformations, which amounts to supposing shared autocorrelation of the coupled components. PARAFAC2 is considered nowadays a complicated variation of the approximate CPD, and many fail to see its area of applications.

Algorithm 13 Fully Alternating LS algorithm for solving the exact coupled CPD problem

INPUT: Multiway arrays $\boldsymbol{\mathcal{T}}_i$, initials factors $\boldsymbol{A}_i$, $\boldsymbol{B}_i$ and $\boldsymbol{C}_i$.

while convergence criterion is not met **do**

$\boldsymbol{C}^* = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{C}_i$

for i from 1 to N **do**

$\boldsymbol{C}_i = \boldsymbol{C}^*$

$\boldsymbol{A}_i = \boldsymbol{T}_{i,(1)} \left(\boldsymbol{B}_i \odot \boldsymbol{C}_i\right) \left(\boldsymbol{B}_i^\mathsf{T}\boldsymbol{B}_i \boxdot \boldsymbol{C}_i^\mathsf{T}\boldsymbol{C}_i\right)^{-1}$

$\boldsymbol{B}_i = \boldsymbol{T}_{i,(2)} \left(\boldsymbol{A}_i \odot \boldsymbol{C}_i\right) \left(\boldsymbol{A}_i^\mathsf{T}\boldsymbol{A}_i \boxdot \boldsymbol{C}_i^\mathsf{T}\boldsymbol{C}_i\right)^{-1}$

$\boldsymbol{C}_i = \boldsymbol{T}_{i,(3)} \left(\boldsymbol{A}_i \odot \boldsymbol{B}_i\right) \left(\boldsymbol{A}_i^\mathsf{T}\boldsymbol{A}_i \boxdot \boldsymbol{B}_i^\mathsf{T}\boldsymbol{B}_i\right)^{-1}$

end for

end while

OUTPUT: estimated factors $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{C}_i$ and $\boldsymbol{C}^*$.

However PARAFAC2 is as of now one of the most flexible multiway models that can take into account subject variability efficiently, and has been used notably to process chromatography data [17]. To cast PARAFAC2 in terms of coupled tensor decomposition is a contribution of this manuscript.

As stated above, PARAFAC2 models intra-subject variability using the assumption that along the mode where variability exists, the autocorrelation of the factors is exactly the same. On the other modes, PARAFAC2 supposes no variability, so that they are exactly the same. Under the subject variability formalism, the PARAFAC2 model can be written as follows:

$$\forall i \in [1, N], \quad \begin{cases} \boldsymbol{\mathcal{T}}_i = \left(\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C}_i\right) \boldsymbol{\Sigma_i}_R + \boldsymbol{\mathcal{E}}_i \\ \boldsymbol{C}_i = \boldsymbol{P}_i \boldsymbol{C}^* \\ \boldsymbol{P}_i^\mathsf{T}\boldsymbol{P}_i = \boldsymbol{I} \end{cases} . \tag{4.8}$$

Matrices $\boldsymbol{P}_i \in \mathbb{R}^{M_i \times M}$ stand for some left orthonormal transformations, and M is the dimension of the latent factor. Noise tensors $\boldsymbol{\mathcal{E}}_i$ are i.i.d., Gaussian, with unit covariance. A first assumption made on the intra-subject variability is that the coupling functions are linear. Moreover, since $\boldsymbol{C}_i^\mathsf{T}\boldsymbol{C}_i = \boldsymbol{C}_j^\mathsf{T}\boldsymbol{C}_j$ for all coupled (i, j), the covariance among the R sources contained in each $\boldsymbol{C}_i$ is shared by all tensors. Conversely, Harshman showed that under these two assumptions, (4.4) is equivalent to considering model (4.8). Finally, the minimal size that a latent variable $\boldsymbol{C}^*$ may have in PARAFAC2 is $R \times R$, since it must have the same number of columns and column rank than all $\boldsymbol{C}_i$. Generically, this implies that M should be greater than R. In practice $M = R$ unless there are further reasons to choose M strictly greater than R, *e.g.* if some physical interpretation can be given to $\boldsymbol{C}^*$ which requires a certain size. In what follows, we suppose $M = R$. This choice is justified in section 4.3 dealing with joint dimensionality reduction.

In PARAFAC2, the left orthonormal transformations $\boldsymbol{P}_i$ cannot be identified. Indeed suppose a set of $\boldsymbol{P}_i$ satisfying (4.8) is known, then given any left orthonormal matrix $\boldsymbol{Q}$ of adequate sizes, $\boldsymbol{Q}\boldsymbol{P}_i$ also satisfies (4.8). In other words, since the invariant in the variability is the autocorrelation of the components, $\boldsymbol{P}_i$ can be found only in the orbit of the set of left orthonormal matrices. Of course the latent factor $\boldsymbol{C}^*$ will also only be identifiable up to a left orthonormal transformation.

An efficient algorithm for solving (4.8) under Gaussian noise was developed when Kiers *et al.* [71] proposed to use PARAFAC2 in order to extract temporal signatures from chromatography data. Indeed variability on the time response, known as retention shifts, is known to be a major drawback for applying CPD to the unmixing of chromatography data recorded through a spectrophotometer (see section 2). To compute PARAFAC2 in an efficient manner, the norms of each component stored in $\boldsymbol{\Sigma_i}$, which are not shared, have to be dealt with. For computational purposes, $\boldsymbol{\Sigma_i}$ is pulled in a fourth way by introducing vectors $\boldsymbol{\lambda}_i$ of size $1 \times R$ as follows:

$$\forall i \in [1, N], \quad \begin{cases} \boldsymbol{\mathcal{T}}_i = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C}_i \otimes \boldsymbol{\lambda_i}) \boldsymbol{\mathcal{I}}_R + \boldsymbol{\mathcal{E}}_i \\ \boldsymbol{C}_i = \boldsymbol{P}_i \boldsymbol{C}^* \\ \boldsymbol{P}_i^\mathsf{T} \boldsymbol{P}_i = \boldsymbol{I} \end{cases} . \tag{4.9}$$

Now it can be seen that (4.9) induces an approximate CPD of a fourth order tensor. Defining $\boldsymbol{\mathcal{G}}_i = \boldsymbol{\mathcal{T}}_i \bullet_3 \boldsymbol{P}_i^\mathsf{T}$, the computation of the PARAFAC2 decomposition of the $\boldsymbol{\mathcal{T}}_i$ amounts to the computation of the approximate CPD of $\boldsymbol{\mathcal{G}} \in \mathbb{R}^K \otimes \mathbb{R}^L \otimes \mathbb{R}^M \otimes \mathbb{R}^N$ which concatenates the third order tensors $\boldsymbol{\mathcal{G}}_i$ on the fourth mode:

$$\boldsymbol{\mathcal{G}} = \left(\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C}^* \otimes \boldsymbol{\Lambda}\right) \boldsymbol{\mathcal{I}}_R + \boldsymbol{\mathcal{E}} \tag{4.10}$$

where $\boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{\lambda}_1 \\ \vdots \\ \boldsymbol{\lambda}_N \end{bmatrix}$ and $\boldsymbol{\mathcal{E}}$ is white if all $\boldsymbol{\mathcal{E}}_i$ were white themselves.

Since the definition of $\boldsymbol{\mathcal{G}}$ depends on coupling matrices $\boldsymbol{P}_i$, those have to be estimated beforehand. With white Gaussian noise, estimating the $\boldsymbol{P}_i$ in (4.9) in the maximum likelihood sense yields the following optimization problem for all i in $[1, N]$:

$$\underset{\boldsymbol{P}_i^\mathsf{T}\boldsymbol{P}_i = \boldsymbol{I}}{\operatorname{argmin}} \|\boldsymbol{T}_{i,(3)} - \boldsymbol{P}_i \boldsymbol{M}_i\|_F^2 \tag{4.11}$$

where $\boldsymbol{M}_i = \boldsymbol{C}^* \left(\boldsymbol{A} \odot \boldsymbol{B} \odot \boldsymbol{\lambda}_i\right)^\mathsf{T}$. Since the Frobenius norm for matrices is related to the trace operator (see Appendix A), minimizing (4.11) is equivalent to maximizing $\operatorname{Tr}\left(\boldsymbol{P}_i \boldsymbol{M}_i \boldsymbol{T}_{i,(3)}^\mathsf{T}\right)$.

By computing the SVD of $\boldsymbol{M}_i \boldsymbol{T}_{i,(3)}^\mathsf{T} = \boldsymbol{U}_i \boldsymbol{\Sigma}_i \boldsymbol{V}_i^\mathsf{T}$, the function to be maximized becomes

$$\operatorname{Tr}\left(\left(\boldsymbol{V}_i^\mathsf{T} \boldsymbol{P}_i \boldsymbol{U}_i\right) \boldsymbol{\Sigma}_i\right) = \sum_{r=1}^{R} [\boldsymbol{V}_i^\mathsf{T} \boldsymbol{P}_i \boldsymbol{U}_i]_{kk} \sigma_k \tag{4.12}$$

where $\boldsymbol{\Sigma}_i = \operatorname{Diag}(\sigma_k)$. The function is maximal if the diagonal values of $\boldsymbol{V}_i^\mathsf{T} \boldsymbol{P}_i \boldsymbol{U}_i$ are as high as possible, and since it is an orthogonal matrix, this yields $\boldsymbol{V}_i^\mathsf{T} \boldsymbol{P}_i \boldsymbol{U}_i = \boldsymbol{I}$, so that

$$\boldsymbol{P}_i = \boldsymbol{V}_i^\mathsf{T} \boldsymbol{U}_i. \tag{4.13}$$

Once the coupling matrices $\boldsymbol{P}_i$ are estimated knowing the other variables, $\boldsymbol{\mathcal{G}}$ can be decomposed using any CPD algorithm, for instance ALS. After this decomposition, coupling are estimated again, and so on. Since the whole optimization process relies on an alternating scheme, the CPD of $\boldsymbol{\mathcal{G}}$ does not need to be precise; rather when using ALS, only one iteration of least squares for each factor should be computed. Algorithm 14 computes PARAFAC2 based on the aproximate CPD of $\boldsymbol{\mathcal{G}}$ computed through ALS, and the estimation of coupling matrices through SVD.

It is noted in [17] that PARAFAC2 is sensitive to the initialization. A good way to initialize PARAFAC2 can be to run N independant PARAFAC models. Actually, since other models introduced in this chapter are also sensitive to initialization, more results on choosing the initialization are introduced in section 4.4.2

Regarding the identifiability of all parameters in PARAFAC2, except for the fact that coupling matrices and the latent factor belong to some equivalence set, nothing is known. This is actually the case for all models introduced in this chapter (except the noiseless exact coupling model). This is clearly a complicated issue. Indeed, suppose that the coupling matrices $\boldsymbol{P}_i$ are known perfectly beforehand. Then computing the PARAFAC2 model for tensors $\boldsymbol{\mathcal{T}}_i$ reduces to computing the approximate CPD of tensor $\boldsymbol{\mathcal{G}}$, and CPD identifiability conditions applied on $\boldsymbol{\mathcal{G}}$ give sufficient conditions for the identifiability of parameters in the PARAFAC2 model. Recall however that for approximate decomposition and thus for PARAFAC2 with known coupling matrices, little is known on the existence of a best approximation and uniqueness of the factors.

4.1.4 Other coupling models

Shift-PARAFAC

In a similar spirit to PARAFAC2, Harshman and Song developed a tensor decomposition model accounting for shifts among coupled factors [61, 66, 67]. A modern formulation of the so-called

Algorithm 14 PARAFAC2-ALS

INPUT: Tensors $\boldsymbol{\mathcal{T}}_i$, initials factors $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}^*$ and $\boldsymbol{\lambda}_i$

while convergence criterion is not met **do**

for $i = 1..N$ **do**

$\boldsymbol{U}_i \boldsymbol{\Sigma_i} \boldsymbol{V}_i^\mathsf{T} = \text{SVD}\left(\boldsymbol{C}^* \left(\boldsymbol{A} \odot \boldsymbol{B} \odot \boldsymbol{\lambda}_i\right)^\mathsf{T} \boldsymbol{T}_{i,(3)}^\mathsf{T}\right)$
$\boldsymbol{P}_i = \boldsymbol{U}_i \boldsymbol{V}_i^\mathsf{T}$
$\boldsymbol{\mathcal{G}}_i = \boldsymbol{\mathcal{T}}_i \bullet_3 \boldsymbol{P}_i^\mathsf{T}$
end for

Define fourth order concatenated tensor $\boldsymbol{\mathcal{G}}$ so that $\boldsymbol{G}_{(4)} = \begin{bmatrix} \text{vec}(\boldsymbol{\mathcal{T}}_1)^\mathsf{T} \\ \vdots \\ \text{vec}(\boldsymbol{\mathcal{T}}_N)^\mathsf{T} \end{bmatrix}$

Compute $\boldsymbol{\mathcal{G}} = \left(\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C}^* \otimes \boldsymbol{\Lambda}\right) \boldsymbol{\mathcal{I}}_R$ using one or few steps of ALS
end while

OUTPUT: estimated factors $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}^*$ and $\boldsymbol{\lambda_i}$, and estimated coupling matrices $\boldsymbol{P}_i$.

Shift-PARAFAC is however due to Morup *et al.* [89], who also developed an efficient algorithm for estimating the parameters of the model.

The idea behind Shift-PARAFAC is that the same process is expressed in all multiway data $\boldsymbol{\mathcal{T}}_i$ but shifted by the following shift operator

$$\tau_i^\delta \left((x_j)_{1 \leqslant j \leqslant N} \right) = \begin{cases} (x_{i-\delta}) & \text{if } N + \delta > i > \delta \\ 0 & \text{elsewhere} \end{cases} . \tag{4.14}$$

This definition of the shift operator differs slightly from the definition given in [89] since the signals are not supposed periodic, and therefore zero-padding is used to fill up unknown values appearing when shifting border coefficients.

Since Shift-PARAFAC in [89] is used for mining the relations in a collection of matrices, let us suppose the N data sets are second-order tensors $\boldsymbol{M}_i \in \mathbb{R}^K \otimes \mathbb{R}^L$. Then the Shift-PARAFAC model can be expressed under the subject variability formalism as follows:

$$\forall i \in [1, N], \quad \begin{cases} \boldsymbol{M}_i = (\boldsymbol{A} \otimes \boldsymbol{B}_i)\, \boldsymbol{\Sigma_i}_R + \boldsymbol{E}_i \\ \boldsymbol{b}_r^{(i)} = \tau^{\delta_{ir}}(\boldsymbol{b}_r^*) \end{cases} \tag{4.15}$$

where $\boldsymbol{B}^* = \left[\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_r^*\right]$ is the latent factor and $\boldsymbol{B}_i = \left[\boldsymbol{b}_1^{(i)}, \ldots, \boldsymbol{b}_r^{(i)}\right]$ are the coupled factors. We denote by $\boldsymbol{\tau}_i$ the shift operator acting on the whole factor $\boldsymbol{B}_i$. In Shift-PARAFAC, the latent factor is identifiable up to any arbitrary shift, since the invariant in the subject variability is the shift.

If δ_{ir} are integers, $\boldsymbol{\tau}_i$ are linear operators. Then similarly to PARAFAC2, the computation of the Shift-PARAFAC can almost be reduced to an unconstrained CPD once the shift operators $\boldsymbol{\tau}_i$ are known. Indeed if $\boldsymbol{G}_i$ is defined by $\boldsymbol{G}_i = \boldsymbol{M}_i \bullet_2 \boldsymbol{\tau}_i^{-1}$, then on their second mode, all $\boldsymbol{G}_i$ feature an observation of $\boldsymbol{B}^*$ truncated on the edges. Otherwise each $\boldsymbol{B}_i$ can be estimated independently in an alternate fashion. Finally, to estimate $\boldsymbol{\tau}_i$, it is possible to modify the coupling relation by using the discrete Fourier transform. Indeed, should matrices $\boldsymbol{M}_i$ be written in the frequency domain, then the shift operator becomes a complex multiplication. More about this method, notably a decomposition algorithm, extensive simulations and a discussion on the uniqueness of the exact Shift-PARAFAC model are found in [89].

Convolutive CPD

Yet another model for modeling subject variability was again proposed by Morup et al. [90]. Again the goal was not solely to account for variability, but also to avoid swamps and degeneracies of the CPD. In Convolutive CPD, shared factors relate to the latent factor through the convolution of the latent components with unknown functions. It is more general and therefore more flexible

than the shift CPD, but on the downside the number of parameters is much greater. Since the Convolutive CPD is one interesting instance of coupled models, it is shortly exposed here although again the whole study of this model is found in [90].

The convolutive CPD is defined as follows:

$$\forall i \in [1, N], \quad \begin{cases} \boldsymbol{M}_i = (\boldsymbol{A} \otimes \boldsymbol{B}_i)\, \boldsymbol{I}_R + \boldsymbol{E}_i \\ \boldsymbol{b}_r^{(i)} = \sum\limits_{k=1}^{K} \mu_k^{(i,r)} \tau^k(\boldsymbol{b}_r^*) \end{cases} \tag{4.16}$$

where $\mu_k^{(i,r)}$ is the coefficient of a convolutive filter for delay k and vector $\boldsymbol{b}_r^{(i)}$, and K is the maximal delay of the filter. Convolutive CPD can be seen as a direct generalization of Shift PARAFAC where $\boldsymbol{B}_i$ is a linear combination of all shifted versions of the latent factor $\boldsymbol{B}^*$.

Similarly to the previous models, it is here possible in theory to reduce the computation problem to the computation of the CPD of some other tensor. Indeed if deconvolution is achievable for each $\boldsymbol{b}_r^{(i)}$, then knowing deconvolution operators yields the CPD of a tensor with third mode factor equals to a matrix of ones. Also there is clearly an indeterminacy on both the coefficients $\mu_k^{(i,r)}$ and the latent factor $\boldsymbol{B}^*$, although again not as easy to express as other indeterminacies introduced above.

Registered PARAFAC

Convolutive CP is a very general model, but a direct consequence of this generality is that more pertinent models can be designed for specific applications. In a recently submitted paper, Rivet *et. al.* have designed a coupling model accounting for differentiable deformation of the x axis so that coupled temporal signals are temporally distorted versions of each other. Though such variability should be found in numerous applications, the one application that gave birth to this model called Registered PARAFAC is the detection of artifacts in EEG [99]. For a discussion on EEG multiway data, see section 2.

A major obstacle to source separation in EEG is indeed the poor signal to noise ratio due to numerous artifacts in the signal. Notably, eye movement is known to generate a signal more powerful than the signal emitted by sources of interest. In [99], a new coupling model called Registered PARAFAC is used to estimate these artifacts in the EEG recordings. Registered PARAFAC takes into account the fact that different artifacts stem from different eye movements which of course are not all identical over time. On the other hand, CPD supposes that all eye movements generate the same temporal response.

The coupling functions used to model temporal distortion are shape-preserving diffeomorphisms f_{ir}, *i.e.* infinitely differentiable functions. Actually, columns of factors $\boldsymbol{B}_i$ are considered as continuous signals in $[0, x_{min}^{(i)}]$, and the coupling is computed in the continuous time domain. Since time is the x variable of the signals, the coupling equations are of the form $\boldsymbol{b}_r^{(i)}(t) = \boldsymbol{b}_r^*(f_{ir}(t))$. Moreover, to simplify the problem, the diffeomorphisms f_{ir} verify

$$\forall x \in [0, x_{max}^{(i)}], \quad \frac{df_{ir}}{dx}(x) > 0, \quad 0 \leqslant f_{ir}(x) \leqslant 1, \quad f_{ir}(0) = 0, \quad f_{ir}(x_{max}^{(i)}) = 1. \tag{4.17}$$

The first two assumptions in (4.17) are necessary to ensure that the signal is not folded, rewinded nor forecasted. An exemple of shape-preserving diffeomorphism is given in Figure 4.2, and Figure 4.3 presents a simple example of a warped signal. Moreover in this context, coupled signals start and end at constant values, and the latent signal is fixed to be in $[0, 1]$, but this does not remove the coupling indeterminacies. To fix the indeterminacies, it is suggested in [99] to choose matrix $\boldsymbol{B}^*$ as the mean of factors $\boldsymbol{B}_i$. The distance used to build the mean of the factors should not be the Euclidian distance, since two factors in the same class of equivalence should have zero distance. An elastic metric may yet be chosen, for instance the Fisher-Rao Riemannian metric [74, 97]. Because under this metric the (Karcher) mean is unique, the latent factor can be uniquely determined from the estimated factors $\boldsymbol{B}_i$. In other words, to simplify the computation of all parameters, in [99] the latent factor is evaluated as the mean of the estimated coupled factors, but the metric used to compute the distances between the estimated factors depends on the estimated coupling functions. Intuitively, this constraint imposes that the coupling functions f_{ir} are centered around the identity operator. This is however not mandatory if working with indeterminacies is not a problem.

Since tensor decomposition models are meant to be computed, it is not sufficient to design a continuous coupling model. Rather, components factors $\boldsymbol{B}_i$ are observed on a discrete grid, and

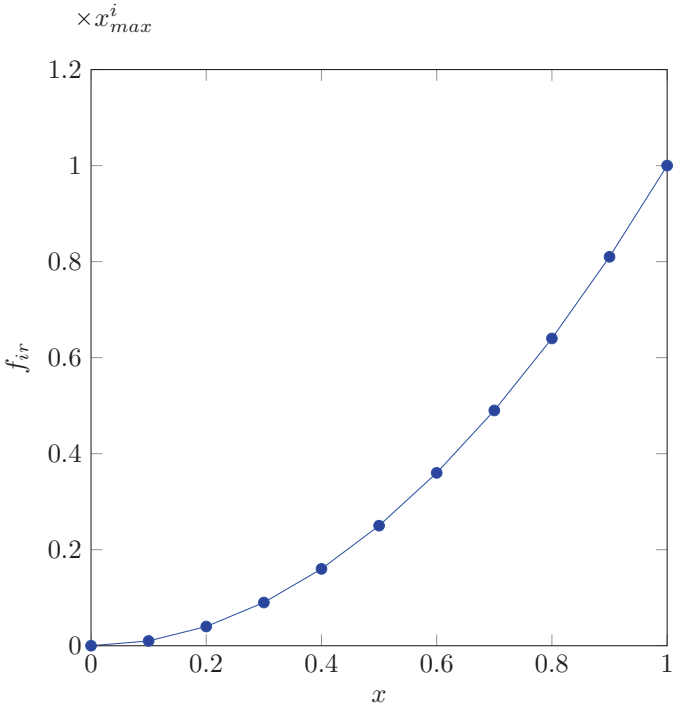


Figure 4.2: An exemple of shape-preserving diffeomorphism. Here $f_{ir}(x) = x^2$.

if an abstract grid is provided for the latent factor $\boldsymbol{B}^*$, coupling functions f_{ir} can be understood as mappings from one grid to another. Thus once values for f_{ir} are known on the grid, a linear relationship can be built between $\boldsymbol{b}_r^{(i)}$ and $\boldsymbol{b}_r^*$ through interpolation:

$$\boldsymbol{b}_r^{(i)} = \boldsymbol{\Gamma}_{ir} \boldsymbol{b}_r^* \tag{4.18}$$

where $\boldsymbol{\Gamma}_{ir}$ is obtained, for instance, by linearly interpolating values of $\boldsymbol{b}_r^*$ on the grid of $\boldsymbol{b}_r^{(i)}$. We do not discuss here potential issues with sampling the continuous signals, but possibly a bound on the derivative of the diffeomorphism could be added to ensure that the Shannon theorem is verified for all $\boldsymbol{b}_r^{(i)}$ if it is verified for $\boldsymbol{b}_r^*$. Working solely in the continuous domain is a perspective to be explored here.

The Registered PARAFAC model is thus defined by the following system:

$$\forall i \in [1, N], \quad \begin{cases} \boldsymbol{M}_i = (\boldsymbol{A} \otimes \boldsymbol{B}_i)\, \boldsymbol{\Sigma}_R + \boldsymbol{E}_i \\ \boldsymbol{b}_r^{(i)} = \boldsymbol{\Gamma}_{ir} \boldsymbol{b}_r^* \end{cases} . \tag{4.19}$$

Since shape-preserving diffeomorphisms are invertible, similarly to PARAFAC2, a third order tensor can be built. Its CPD yields factors $\boldsymbol{A}$, $\boldsymbol{B}^*$ and scales $\boldsymbol{\Sigma}_i$. However in [99], the estimation of the diffeomorphisms requires $\boldsymbol{B}_i$ to be known since a full alternating algorithm is used. The estimation of the f_{ir} relies on the Fisher-Rao metric so that the Karcher mean constraint of the latent factor is satisfied at all points in the optimization process. The complete decomposition algorithm is provided in [99], but since the focus here is on the models rather than complex optimization algorithm, it is not reported here.

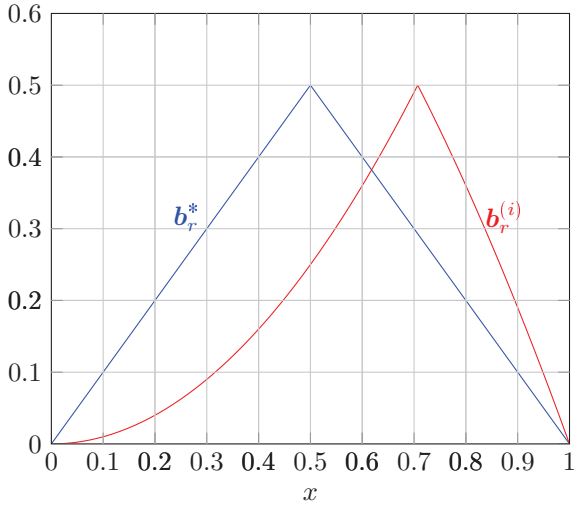


Figure 4.3: A triangle signal $\boldsymbol{b}_r^*$ warped into $\boldsymbol{b}_r^{(i)}$ using the shape-preserving diffeomorphism $f_{ir}(x) = x^2$.

4.2 Flexible Data fusion

As demonstrated by the previous section, various tensor decomposition models exist that account for some variability among subjects, or in other words, that design coupling functions between related factors in multiple CPDs. However, a general framework that encompass approximate couplings, transformed couplings and non-Gaussian noise distributions is yet to be provided. The following section introduces a Maximum A Posteriori (MAP) estimator to optimally decompose coupled tensors when all these scenarios are intertwined. This sheds a new light on previously presented coupled CPD models. Further, decomposition algorithms for noisy coupling models are designed, and optimal performance under Gaussian noise are studied through Cramér-Rao bounds for coupled models. Finally, we provide an application to the fluorescence/NMR data introduced in section 2.

4.2.1 Bayesian formulation

Consider two arrays of noisy measurements, $\boldsymbol{\mathcal{T}}_1$ and $\boldsymbol{\mathcal{T}}_2$, which can be tensors of possibly different orders and dimensions. Arrays $\boldsymbol{\mathcal{T}}_1$ and $\boldsymbol{\mathcal{T}}_2$ are related to two parametric models characterized by parameter vectors $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, respectively. For instance, $\boldsymbol{\mathcal{T}}_1$ may follow an array normal distribution with diagonal unit covariance and centered on a small rank tensor $(\boldsymbol{A}_1 \otimes \boldsymbol{B}_1 \otimes \boldsymbol{C}_1)\boldsymbol{\mathcal{I}}_R$. If $\boldsymbol{\mathcal{T}}_2$ also follows an approximate CPD model, then vectors of parameters are stacks of vectorized factors of the CPDs, that is:

$$\boldsymbol{\theta}_i = \begin{bmatrix} \text{vec}\,(\boldsymbol{A}_i) \\ \text{vec}\,(\boldsymbol{B}_i) \\ \text{vec}\,(\boldsymbol{C}_i) \end{bmatrix} \tag{4.20}$$

for $i \in \{1; 2\}$.

Some parameters contained in each $\boldsymbol{\theta}_i$ can be shared or related through some known probability distribution. Then the data sets $\boldsymbol{\mathcal{T}}_1$ and $\boldsymbol{\mathcal{T}}_2$ are said to be coupled. In what follows, the goal is to find an estimator of all parameters derived from Bayesian statistical tools that optimally uses the fact that some parameters are linked. The case of more than two coupled data sets brings some difficult questions that will be quickly over-viewed in section 4.4.1.

In the case where $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are not related, they may be obtained separately from each data set. On the other hand, if parameters are somehow linked, there is some gain to estimate them simultaneously as shown below.

Here we consider that the coupling between $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ is flexible. To formalize this we assume that the pair $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ is random and that we have at our disposal a joint probability density function (PDF) $p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$. This is to be contrasted with models discussed above for which the relations between factors are deterministic.

Maximum *a posteriori* estimator since the pair $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ is random, a MAP setting[5] is adopted. In this setting, the approximation problem is defined as

$$\begin{aligned}\arg\max_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2} p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2|\boldsymbol{\mathcal{T}}_1, \boldsymbol{\mathcal{T}}_2) &= \arg\max_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2} p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\mathcal{T}}_1, \boldsymbol{\mathcal{T}}_2) \\ &= \arg\min_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2} \Upsilon(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)\end{aligned}$$

where $\Upsilon(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = -\log p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\mathcal{T}}_1, \boldsymbol{\mathcal{T}}_2)$. Conditioning on the parameters leads to a cost function that can be decomposed into a joint data likelihood term plus a term involving the coupling:

$$\Upsilon(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = -\log p(\boldsymbol{\mathcal{T}}_1, \boldsymbol{\mathcal{T}}_2|\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) - \log p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) \tag{4.21}$$

Under the following simplifying hypotheses underlying the Bayesian approach, the MAP estimator is given as the minimizer of a three-term cost function :

$$\arg\min_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2} \Upsilon(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \arg\min_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2} \left[-\log p(\boldsymbol{\mathcal{T}}_1|\boldsymbol{\theta}_1) - \log p(\boldsymbol{\mathcal{T}}_2|\boldsymbol{\theta}_2) - \log p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)\right] \tag{4.22}$$

H1 *Conditional independence of the data*: the data arrays $\boldsymbol{\mathcal{T}}_1$ and $\boldsymbol{\mathcal{T}}_2$ are independent of $\boldsymbol{\theta}_2$ and $\boldsymbol{\theta}_1$ respectively, if they are conditioned on $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$. We suppose also that they are conditionally independent on each other. This results in the following $p(\boldsymbol{\mathcal{T}}_1|\boldsymbol{\mathcal{T}}_2, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = p(\boldsymbol{\mathcal{T}}_1|\boldsymbol{\theta}_1)$ and $p(\boldsymbol{\mathcal{T}}_2|\boldsymbol{\mathcal{T}}_1, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = p(\boldsymbol{\mathcal{T}}_2|\boldsymbol{\theta}_2)$.

H2 *Independence of uncoupled parameters*: all parameters except the coupled parameters are independent. In the joint CPD case, this means that $p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = p(\boldsymbol{C}_1, \boldsymbol{C}_2)p(\boldsymbol{A}_1)p(\boldsymbol{A}_2)p(\boldsymbol{B}_1)p(\boldsymbol{B}_2)$ if coupling exists only on the third mode.

H3 *On the priors*: trivially, the joint distribution of the coupled parameters, *e.g.* $p(\boldsymbol{C}_1, \boldsymbol{C}_2)$ needs to be known or, at least, one of the conditional distributions, namely $p(\boldsymbol{C}_1|\boldsymbol{C}_2)$ or $p(\boldsymbol{C}_2|\boldsymbol{C}_1)$. The marginal priors on the uncoupled and on the conditioning parameters are assumed either to be known or to be flat on some domain of definition.

H4 *Likelihoods*: the conditional probabilities (or likelihoods) $p(\boldsymbol{\mathcal{T}}_1|\boldsymbol{\theta}_1)$ and $p(\boldsymbol{\mathcal{T}}_2|\boldsymbol{\theta}_2)$ are known, at least up to a scale parameter. In a MAP setting, this indirectly sets the weights which will be given to each data array in $\Upsilon(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$.

Hypotheses H3 and H4 are assumed so that all terms in this cost function are defined. H2 is assumed so that the probabilistic dependence of the parameters defines the coupling between various latent models.

The framework presented above contains as specific cases the following:

- *Hybrid estimation:* if we consider that the conditional distribution $p(\boldsymbol{C}_1|\boldsymbol{C}_2)$ is known and that $\boldsymbol{C}_2$ is deterministic and defined in a domain $\boldsymbol{\Omega}_{\boldsymbol{C}_2}$, then we have to minimize $\Upsilon_H(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = -\log p(\boldsymbol{\mathcal{T}}_1|\boldsymbol{\theta}_1) - \log p(\boldsymbol{\mathcal{T}}_2|\boldsymbol{\theta}_2) - \log p(\boldsymbol{C}_1|\boldsymbol{C}_2)$ subject to $\boldsymbol{C}_2 \in \boldsymbol{\Omega}_{\boldsymbol{C}_2}$, which is a hybrid (random/deterministic) coupled approximation problem. This is the framework implicitly considered in [102].

- *Exact coupled CPD:* the exact coupled CPD can be obtained by setting a Dirac's delta prior $p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \delta(\boldsymbol{C}_1 - \boldsymbol{C}_2)$. The MAP problem becomes a MLE problem with equality constraints

$$\begin{array}{cc} \text{maximize} & \log p(\boldsymbol{\mathcal{T}}_1|\boldsymbol{\theta}_1) + \log p(\boldsymbol{\mathcal{T}}_2|\boldsymbol{\theta}_2) \\ \text{with respect to (w.r.t.)} & \boldsymbol{\theta}_1,\, \boldsymbol{\theta}_2 \\ \text{subject to} & \boldsymbol{C}_1 = \boldsymbol{C}_2 \end{array} \tag{4.23}$$

  which corresponds to the coupled approximation framework with general cost functions. Different versions of this approach are proposed in [54, 103, 124]. Under the assumption of Gaussian likelihoods, 4.23 is exactly the exact coupled CPD.

- *Exactly coupled decompositions:* additionally to exact coupling of one of the factors, if the data sets are not corrupted by noise, the likelihoods are also products of delta functions and we have to solve an exactly coupled decomposition problem [104], [105].

[5] We could also consider a minimum mean squared error setting but then we would need to evaluate $p(\boldsymbol{\mathcal{T}}_1, \boldsymbol{\mathcal{T}}_2)$, which is typically cumbersome.

This Bayesian approach provides a theoretical justification to coupled model (4.5). Indeed, knowing the joint probability distributions $p(\theta_1, \theta_2)$ and the noise distributions on the CP models yields an optimal estimator of the factors in the maximum *a posteriori* sense. Therefore, when designing a flexible coupled CPD accounting for subject variability, what matters is to determine the distributions $\mathcal{D}_i$ for all coupled factors and the noise distribution on the data $\mathcal{D}_i^{(\boldsymbol{\mathcal{T}})}$. If these distributions admit density functions, then the framework developed in this section can be applied in a straightforward manner.

In the next section we give many examples of joint decomposition problems and their MAP or hybrid MAP/MLE objective functions.

4.2.2 Examples

In what follows we consider that the parametric models underlying the data arrays are two approximate CPD models of rank R_1 and R_2 respectively. We consider that the coupling occurs between matrices $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$. We illustrate this framework with three different examples: general joint Gaussian, hybrid Gaussian and non Gaussian models for the parameters. This section is largely inspired from [21, 22]. The problem of merging data sets with different sampling rates is tackled solely in the research paper, while here flexible coupling models are applied to the chemometrics data set introduced in section 2.1.2.

Joint Gaussian modeling

A general joint Gaussian model including coupled and uncoupled variables is given by the following expression:

$$\boldsymbol{M} \begin{bmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \end{bmatrix} = \boldsymbol{\Sigma}\boldsymbol{u} + \boldsymbol{\mu} \tag{4.24}$$

where $\boldsymbol{M}$ is a matrix defining the structural relations between variables, $\boldsymbol{u}$ is a white Gaussian vector with zero mean and unit variances, $\boldsymbol{\Sigma}$ is a diagonal matrix of standard deviations and $\boldsymbol{\mu}$ is a constant vector. A condition for the pair $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ to define a joint Gaussian vector is the left invertibility of $\boldsymbol{M}$. Under this condition we have

$$\begin{bmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \end{bmatrix} \sim \mathcal{N}\left({}^{\dagger}\boldsymbol{M}\boldsymbol{\mu}, \boldsymbol{R}\right) \tag{4.25}$$

where $\boldsymbol{R} = ({}^{\dagger}\boldsymbol{M})\boldsymbol{\Sigma}\boldsymbol{\Sigma}({}^{\dagger}\boldsymbol{M}^{\mathsf{T}})$ is the covariance matrix of the joint vector.

Assume that the CPD models $\boldsymbol{\mathcal{T}}_1$ and $\boldsymbol{\mathcal{T}}_2$ are measured with zero mean independent additive Gaussian noise, with variances σ_1^2 and σ_2^2, respectively. We have the following MAP objective function:

$$\Upsilon\left(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\right) = \frac{1}{\sigma_1^2}\|\boldsymbol{\mathcal{T}}_1 - (\boldsymbol{A}_1 \otimes \boldsymbol{B}_1 \otimes \boldsymbol{C}_1)\boldsymbol{\mathcal{I}}_{R_1}\|_F^2 + \frac{1}{\sigma_2^2}\|\boldsymbol{\mathcal{T}}_2 - (\boldsymbol{A}_2 \otimes \boldsymbol{B}_2 \otimes \boldsymbol{C}_2)\boldsymbol{\mathcal{I}}_{R_2}\|_F^2 + \left\| \begin{bmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \end{bmatrix} - {}^{\dagger}\boldsymbol{M}\boldsymbol{\mu}\right\|_{\boldsymbol{R}}^2 \tag{4.26}$$

where $\|\cdot\|_{\boldsymbol{R}}$ is the $\boldsymbol{R}$-weighted Frobenius norm. Below is one instance of this approach.

Example 6 *A usual problem in multimodal data fusion is that some components are not present in all modalities, thus we have some components which are shared and some which are specific to each modality. Suppose* $\boldsymbol{\theta}_1 = \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{c}_1 \end{bmatrix}$ *and* $\boldsymbol{\theta}_2 = \begin{bmatrix} \boldsymbol{u}_2 \\ \boldsymbol{c}_2 \end{bmatrix}$*, where* $\boldsymbol{c}_1$ *and* $\boldsymbol{c}_2$ *are coupled, whereas* $\boldsymbol{u}_1$ *and* $\boldsymbol{u}_2$ *are not.*

Let us build a simple coupling between $\boldsymbol{c}_1$ *and* $\boldsymbol{c}_2$*, that is the two random variables have the same mean* $\mathbb{E}[\boldsymbol{c}_1] = \mathbb{E}[\boldsymbol{c}_2]$*. Further supposing zero mean marginal Gaussian priors, we may describe the probability densities of parameters as follows:*

$$\boldsymbol{u}_1 = \boldsymbol{\Sigma}_1^u \boldsymbol{z}_1^u, \quad \boldsymbol{u}_2 = \boldsymbol{\Sigma}_2^u \boldsymbol{z}_2^u \tag{4.27}$$

$$\boldsymbol{c}_1 - \boldsymbol{c}_2 = \boldsymbol{\Sigma}_1^c \boldsymbol{z}_1^c, \quad \boldsymbol{c}_2 = \boldsymbol{\Sigma}_2^c \boldsymbol{z}_2^c \tag{4.28}$$

where $\boldsymbol{z}_1^u$*,* $\boldsymbol{z}_2^u$*,* $\boldsymbol{z}_1^c$ *and* $\boldsymbol{z}_2^c$ *are zero-mean unit variance independent Gaussian random variables, and where* $\boldsymbol{\Sigma}_1^u$*,* $\boldsymbol{\Sigma}_2^u$*,* $\boldsymbol{\Sigma}_1^c$ *and* $\boldsymbol{\Sigma}_2^c$ *define corresponding covariance matrices. Defining block-matrices*

$$\boldsymbol{M} = \left[\begin{array}{c|c|c|c} \boldsymbol{I} & 0 & 0 & 0 \\ \hline 0 & \boldsymbol{I} & 0 & -\boldsymbol{I} \\ \hline 0 & 0 & \boldsymbol{I} & 0 \\ \hline 0 & 0 & 0 & \boldsymbol{I} \end{array}\right], \quad \boldsymbol{\Sigma} = \left[\begin{array}{c|c|c|c} \boldsymbol{\Sigma}_1^u & 0 & 0 & 0 \\ \hline 0 & \boldsymbol{\Sigma}_1^c & 0 & 0 \\ \hline 0 & 0 & \boldsymbol{\Sigma}_2^u & 0 \\ \hline 0 & 0 & 0 & \boldsymbol{\Sigma}_2^c \end{array}\right] \tag{4.29}$$

then equations (4.27-4.28) can be grouped in a unique block equation of the form:

$$\boldsymbol{M}\begin{bmatrix}\boldsymbol{u}_1\\ \boldsymbol{c}_1\\ \boldsymbol{u}_2\\ \boldsymbol{c}_2\end{bmatrix} = \boldsymbol{\Sigma}\begin{bmatrix}\boldsymbol{z}_1^u\\ \boldsymbol{z}_1^c\\ \boldsymbol{z}_2^u\\ \boldsymbol{z}_2^c\end{bmatrix}. \tag{4.30}$$

An interesting point is that to obtain a left-invertible mixing matrix $\boldsymbol{M}$, it is necessary to give enough information on the distributions of all parameters. Since the distribution of $\boldsymbol{c}_1 - \boldsymbol{c}_2$ is known, it is then sufficient to provide only the prior distribution of one of the two vector of parameters $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$. However in practice, prior distributions on all factors may not be available. Rather, it seems natural to only design the conditional coupling relationship $p(\boldsymbol{c}_1|\boldsymbol{c}_2)$, which sole knowledge does not provide a left invertible mixing matrix. The next paragraph tackles this issue using an hybrid deterministic-Bayesian framework.

Hybrid Gaussian coupling

When no prior information is given on some parameters, full joint Gaussian modeling is inadequate, since we cannot define means and variances of the model. Instead, we consider these parameters as deterministic, while the other parameters are still stochastic with Gaussian priors. This model is called *hybrid Gaussian*. This particular case is of outmost importance since most applications will fall in this category of coupling. Indeed, it covers the scenario where only one factor, say $\boldsymbol{C}$, is coupled to another, say $\boldsymbol{C}_2$, with transformation matrices and independent and identically distributed (i.i.d.) Gaussian additive noise:

$$\boldsymbol{H}_1\boldsymbol{C}_1 = \boldsymbol{H}_2\boldsymbol{C}_2 + \boldsymbol{\Gamma}, \quad \Gamma_{ij} \sim \mathcal{N}(0, \sigma_c^2) \tag{4.31}$$

for some transformation matrices $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$. If relation (4.31) is the only known relationship between parameters, it means that only the conditional probability $p\left(\boldsymbol{C}_1|\boldsymbol{C}_2\right)$ is known, and that $\boldsymbol{C}_2$ is either deterministic or has a flat non-informative prior. This scenario is exactly the theoretical framework to which exact coupled CPD, PARAFAC2, Shift-CP and other previously introduced coupled decomposition models belong.

The hybrid coupling model cannot be written with zero noise variance if the transformation matrices are tall (*i.e.* more rows than columns), as the set of feasible parameters may be reduced to the trivial set. Indeed having tall coupling matrices means that the coupling model has more equations than parameters. Using zero noise variance as a working hypothesis as in [4] leads to a bias in the hybrid Gaussian coupling estimation setting when transformation matrices are tall. Tall coupling matrices may occur when they stand for interpolation operators, as is lengthly discussed in [22].

Even more critical is the norm of the coupled factors. Without proper normalization, the CP decomposition is invariant w.r.t. the norm of columns of $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ as it can be incorporated in the other factors. This means that the variance of $\boldsymbol{\Gamma}$ is not well defined in (4.31). It is necessary to add the information on the norm of columns of the coupled factors in the hybrid model (4.31), or to consider the coupling variance as an unknown parameter. This will also impact algorithms designed in Section 4.2.3.

Non Gaussian conditional coupling

Non trivial couplings expressing similarity between the factors $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ can be addressed by considering that $\boldsymbol{C}_2$ is deterministic and that the coupling is given by a conditional non Gaussian distribution, $p\left(\boldsymbol{C}_1|\boldsymbol{C}_2\right)$.

Impulsive additive coupling As a first example, we can assume that each element in $\boldsymbol{C}_1$ is a version of $\boldsymbol{C}_2$ corrupted by i.i.d. impulsive noise:

$$\boldsymbol{C}_1 = \boldsymbol{C}_2 + \Gamma \tag{4.32}$$

where Γ_{ir} follows a Laplacian distribution $p(\Gamma_{ir}) = (1/2\delta_c)\exp(-|\Gamma_{ir}|/\delta_c)$ with scale parameter δ_c or a Cauchy distribution $p(\Gamma_{ir}) = 1/\{\pi\delta_c[1 + (W_{ir}/\delta_c)^2]\}$. The objective functions to be minimized

are then

$$\Upsilon\left(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2\right) = -\log p(\boldsymbol{\mathcal{T}}_1|\boldsymbol{\theta}_1) - \log p(\boldsymbol{\mathcal{T}}_2|\boldsymbol{\theta}_2) + \begin{cases} +(1/2\delta_c)\|\boldsymbol{C}_1 - \boldsymbol{C}_2\|_1 & \text{(Laplace)} \\ -\sum_{ir}^{MR} \log\{1 + [(C_1^{ir} - C_2^{ir})/\delta_c]^2\} & \text{(Cauchy)} \end{cases} \tag{4.33}$$

where $\|\cdot\|_1$ is taken coefficient-wise. The first penalty was considered in [102] in a collective matrix factorization context. Both cost functions imply a small number of large discrepancies between $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$.

Positive general coupling When $\boldsymbol{C}_1 \succeq 0$ and $\boldsymbol{C}_2 \succeq 0$, an additive noise in the coupling may not be well suited to model noise, since, to ensure C_1^{ir} are positive, the support of the additive term has to depend on the values of C_2^{ir}, which is not a realistic assumption on a perturbation term. Therefore, other alternatives naturally ensuring positiveness can be considered. For example, the Tweedie's distribution [69][Jorgensen1992] which contains Poisson, Gamma and inverse-Gaussian distributions as special cases (the Gaussian distribution is a limiting special case). In general, the PDF of the Tweedie's distribution has no analytical form, thus we cannot directly use it to write down a coupling term in the MAP objective function. However, if we consider that the coupling between $\boldsymbol{C}_1 \succeq 0$ and $\boldsymbol{C}_2 \succeq 0$ is strong (dispersion ϕ_c is small), then a saddle point approximation can be used [69]:

$$p(C_1^{ir}|C_2^{ir}) \approx (2\pi\phi_c(C_1^{ir})^{\beta_c})^{-1/2} \exp[-d_{\beta_c}(C_1^{ir}|C_2^{ir})/\phi_c] \tag{4.34}$$

where β_c is a shape parameter ($\beta_c = 1, 2, 3$ for the Poisson, Gamma and inverse-Gaussian distributions respectively) and d_{β_c} is the beta divergence [125]:

$$d_{\beta_c}(C_1^{ir}|C_2^{ir}) = \frac{(C_2^{ir})^{1-\beta_c}}{\kappa(\beta_c)}\left[(C_1^{ir})^{2-\beta_c}(C_2^{ir})^{\beta_c-1} - C_1^{ir}(2-\beta_c) + C_2^{ir}(1-\beta_c)\right] \tag{4.35}$$

where $\kappa(\beta_c) = (1-\beta_c)(2-\beta_c)$. Under this conditional distribution, assuming that all factors in the CPD are positive and that the data distributions are also of Tweedie type with shapes β_1 and β_2 and dispersions ϕ_1 and ϕ_2, we have the following hybrid MAP/MLE objective:

$$\begin{aligned} \Upsilon(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2) = & \frac{1}{\phi_1}\left[\sum_{klm}^{K_1L_1M} d_{\beta_1}(\mathcal{T}_{1_{klm}}|\mathcal{X}1_{klm})\right] + \\ & \frac{1}{\phi_2}\left[\sum_{klm}^{K_2L_2M} d_{\beta_2}(\mathcal{T}_{2_{klm}}|\mathcal{X}_{2_{klm}})\right] + \\ & \sum_{ir}^{MR}\left[(\beta_c/2)\log\left(C_1^{ir}\right) + (1/\phi_c)d_{\beta_c}(C_1^{ir}|C_2^{ir})\right] \end{aligned} \quad . \tag{4.36}$$

4.2.3 Algorithms

To compute the maximum a posteriori estimator of the factors in flexible coupled CPD, this section introduces ALS-based algorithms. ALS can be used when the measurement noise and the coupling model are jointly Gaussian, since the unconstrained uncoupled cost function is then quadratic in each factor. If the noise is not Gaussian, some gradient-based algorithms are detailed in [22].

First we provide Algorithm 15 for the joint Gaussian coupling model, which is the most general form of coupled CPD when all distributions are Gaussian. Let us rewrite the cost function in a more convenient manner by supposing the mean $\boldsymbol{\mu}$ is zero. Then $\boldsymbol{M\theta} = \boldsymbol{\Sigma u}$, and

$$\boldsymbol{\Sigma}^{-1}\boldsymbol{M\theta} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{4.37}$$

and let us use the following notation: $\boldsymbol{R}^{\frac{1}{2}} = \boldsymbol{\Sigma}^{-1}\boldsymbol{M}$. Then the maximum a priori estimator with respect to variable $\boldsymbol{A}_1$ is given by

$$\underset{\boldsymbol{A}_1}{\operatorname{argmin}} \frac{1}{\sigma_1^2}\|\boldsymbol{T}_{1,(1)} - \boldsymbol{A}_1\boldsymbol{N}_1^{\mathsf{T}}\|_F^2 + \frac{1}{\sigma_2^2}\|\boldsymbol{T}_{2,(1)} - \boldsymbol{A}_2\boldsymbol{N}_2^{\mathsf{T}}\|_F^2 + \frac{1}{\sigma_c^2}\|\boldsymbol{R}^{\frac{1}{2}}\begin{bmatrix} \operatorname{vec}(\boldsymbol{A}_1) \\ \boldsymbol{\theta}_{\overline{\boldsymbol{A}_1}} \end{bmatrix}\|_F^2 \tag{4.38}$$

where $\boldsymbol{N}_i = \boldsymbol{B}_i \odot \boldsymbol{C}_i$ and $\boldsymbol{\theta}_{\overline{\boldsymbol{A}_1}}$ is the part of $\boldsymbol{\theta}$ that does not contain variables $\operatorname{vec}(\boldsymbol{A}_1)$. Now to compute the derivative of the cost function with respect to $\operatorname{vec}(\boldsymbol{A}_1)$, it is necessary to compute the

derivative of $\|\boldsymbol{R}^{\frac{1}{2}}\begin{bmatrix}\text{vec}\,(\boldsymbol{A}_1)\\ \boldsymbol{\theta}_{\overline{A_1}}\end{bmatrix}\|_F^2$. This derivative features the columns of $\boldsymbol{R}^{\frac{1}{2}}$ which are multiplied by variables in $\text{vec}\,(\boldsymbol{A}_1)$, denoted by $\boldsymbol{R}^{\frac{1}{2}}_{\boldsymbol{A}_1}$. Then

$$\frac{\partial}{\partial \text{vec}\,(\boldsymbol{A}_1)}\boldsymbol{R}^{\frac{1}{2}}\begin{bmatrix}\text{vec}\,(\boldsymbol{A}_1)\\ \boldsymbol{\theta}_{\overline{\boldsymbol{A}_1}}\end{bmatrix} = \boldsymbol{R}^{\frac{1}{2}}_{\boldsymbol{A}_1}. \tag{4.39}$$

Here $\boldsymbol{R}^{\frac{1}{2}}_{\boldsymbol{A}_1}$ is the upper block of $\boldsymbol{R}^{\frac{1}{2}}$, but $\boldsymbol{R}^{\frac{1}{2}}_{\boldsymbol{B}_1}$ for instance corresponds to another block in $\boldsymbol{R}^{\frac{1}{2}}$. Finally, the update rule for $\boldsymbol{A}_1$ is obtained by setting to zero the following derivative of the cost function:

$$\left[\frac{1}{\sigma_1^2}\left(\boldsymbol{I}\otimes\boldsymbol{N}_1^\mathsf{T}\boldsymbol{N}_1\right)+\frac{1}{\sigma_c^2}\left[\boldsymbol{R}^{\frac{-\mathsf{T}}{2}}_{\boldsymbol{A}_1}\boldsymbol{R}^{\frac{1}{2}}\right]_{\boldsymbol{A}_1}\right]\text{vec}\,(\boldsymbol{A}_1) = \frac{1}{\sigma_1^2}\text{vec}\left(\boldsymbol{T}_{1,(1)}\boldsymbol{N}_1\right)-\frac{1}{\sigma_c^2}\left[\boldsymbol{R}^{\frac{-\mathsf{T}}{2}}_{\boldsymbol{A}_1}\boldsymbol{R}^{\frac{1}{2}}\right]_{\overline{\boldsymbol{A}_1}}\boldsymbol{\theta}_{\overline{\boldsymbol{A}_1}} \tag{4.40}$$

where for a matrix $\boldsymbol{Q}$, $[\boldsymbol{Q}]_{\boldsymbol{A}_1}$ is the submatrix formed by columns corresponding to variables $\boldsymbol{A}_1$ in $\boldsymbol{\theta}$, similarly to how $\boldsymbol{R}^{\frac{1}{2}}_{\boldsymbol{A}_1}$ is defined. This update rule is used for every factors in an alternate fashion, as described in Algorithm 15.

Algorithm 15 ALS algorithm for solving the joint Gaussian coupling problem

INPUT: Multiway arrays $\boldsymbol{\mathcal{T}}_i$, initials factors $\boldsymbol{A}_i$, $\boldsymbol{B}_i$ and $\boldsymbol{C}_i$, coupling matrix $\boldsymbol{M}$ and coupling noise variance $\boldsymbol{\Sigma}$.
while convergence criterion is not met **do**
  Update $\boldsymbol{A}_1$ by solving linear system (4.40).
  Similarly update $\boldsymbol{A}_2, \boldsymbol{B}_1.\boldsymbol{B}_2, \boldsymbol{C}_1, \boldsymbol{C}_2$.
end while
OUTPUT: estimated factors $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{C}_i$.

If specific constraints apply on the coupling functions, then more dedicated and efficient algorithms can be designed that account for the nature of coupling functions. Algorithms 12,14 are examples of ALS-based algorithms that use the specific structure of the coupling model in PARAFAC2 and Exact coupled CPD. Below an algorithm for computing the Hybrid Gaussian coupling model is developed in Algorithms 16 and 17. Using a full alternating method or a joint estimation of the coupled factors again leads to two types of algorithms, even when only two data set are coupled. ALS algorithms solving the hybrid coupling problem are based on the resolution of the system obtained when differentiating (4.22) with respect to $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$:

$$\begin{cases}\boldsymbol{H}_1^\mathsf{T}\boldsymbol{H}_1\boldsymbol{C}_1+\boldsymbol{C}_1\left(\boldsymbol{A}_1^\mathsf{T}\boldsymbol{A}_1\boxdot\boldsymbol{B}_1^\mathsf{T}\boldsymbol{B}_1\right)-\boldsymbol{H}_1^\mathsf{T}\boldsymbol{H}_2\boldsymbol{C}_2=\boldsymbol{T}_{1,(3)}\left(\boldsymbol{A}_1\odot\boldsymbol{B}_1\right)\\ \boldsymbol{H}_2^\mathsf{T}\boldsymbol{H}_2\boldsymbol{C}_2+\boldsymbol{C}_2\left(\boldsymbol{A}_2^\mathsf{T}\boldsymbol{A}_2\boxdot\boldsymbol{B}_2^\mathsf{T}\boldsymbol{B}_2\right)-\boldsymbol{H}_2^\mathsf{T}\boldsymbol{H}_1\boldsymbol{C}_1=\boldsymbol{T}_{2,(3)}\left(\boldsymbol{A}_2\odot\boldsymbol{B}_2\right)\end{cases}. \tag{4.41}$$

If both $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ are updated simultaneously by solving (4.41), then Algorithm 16 is obtained, whereas an alternated update of $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ leads to Algorithm 17

4.2.4 Normalization of the noisy coupling

A fact that is mostly silenced in this manuscript is that the factors in the CPD can be found only up to a scaling among each others as stated in section 1.4.1. Typically this indeterminacy is not an issue since either only the relative values of the components have physical interpretation, or additional information on the norm of factors is available and used to scale the factors at each step of any iterative decomposition algorithm.

However this fact cannot be ignored when designing an hybrid Gaussian coupling model. Let us suppose two factors $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ are linked through the following equation:

$$\begin{cases}\boldsymbol{C}_1=\boldsymbol{C}_2+\boldsymbol{\Gamma}\\ \boldsymbol{\Gamma}\sim\mathcal{MN}\left(\boldsymbol{0},\frac{1}{\sigma_c^2}\boldsymbol{I}\otimes\boldsymbol{I}\right)\end{cases} \tag{4.46}$$

for some positive real σ_c. Moreover suppose data tensors $\boldsymbol{\mathcal{T}}_i$ follow an approximate CPD with diagonal noise covariance $\frac{1}{\sigma_i^2}\boldsymbol{I}\otimes\boldsymbol{I}\otimes\boldsymbol{I}$. Then the same MAP estimator can be obtained as in 4.22

Algorithm 16 Simultaneous ALS algorithm for the hybrid Gaussian coupled model (for unitary columns in $\boldsymbol{C}_1$)

INPUT: $\boldsymbol{\mathcal{T}}_i$, σ_c, σ_i, initial factors $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{C}_i$, coupling matrices $\boldsymbol{H}_i$.
Apply two uncoupled ALS to obtain $\boldsymbol{A}_1$, $\boldsymbol{B}_1$, $\boldsymbol{C}_1$, $\boldsymbol{A}_2$, $\boldsymbol{B}_2$ and $\boldsymbol{C}_2$.
Permute initialization factors columns to have minimal $\|\boldsymbol{H}_1\boldsymbol{C}_1 - \boldsymbol{H}_2\boldsymbol{C}_2\|_F^2$.
while convergence criterion is not met **do**
  for i=1:2 **do**
    $\boldsymbol{A}_i = \boldsymbol{T}_{i,(1)} \left(\boldsymbol{B}_i \odot \boldsymbol{C}_i\right) \left(\boldsymbol{B}_i^\mathsf{T}\boldsymbol{B}_i \boxdot \boldsymbol{C}_i^\mathsf{T}\boldsymbol{C}_i\right)^{-1}$
    $\boldsymbol{B}_i = \boldsymbol{T}_{i,(2)} \left(\boldsymbol{A}_i \odot \boldsymbol{C}_i\right) \left(\boldsymbol{A}_i^\mathsf{T}\boldsymbol{A}_i \boxdot \boldsymbol{C}_i^\mathsf{T}\boldsymbol{C}_i\right)^{-1}$
  end for
  Update $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ by solving (4.41) using the least squares solution to the following linear system:

$$\boldsymbol{G}\,\mathrm{vec}([\boldsymbol{C}_1;\boldsymbol{C}_2]) = \mathrm{vec}\left(\left[\frac{1}{\sigma_1^2}\boldsymbol{T}_{\mathbf{1},(\mathbf{3})}\left(\boldsymbol{A}_1^\mathsf{T}\boldsymbol{A}_1 \boxdot \boldsymbol{B}_1^\mathsf{T}\boldsymbol{B}_1\right); \frac{1}{\sigma_2^2}\boldsymbol{T}_{\mathbf{2},(\mathbf{3})}\left(\boldsymbol{A}_2^\mathsf{T}\boldsymbol{A}_2 \boxdot \boldsymbol{B}_2^\mathsf{T}\boldsymbol{B}_2\right)\right]\right) \tag{4.42}$$

  where $\boldsymbol{G} = [\boldsymbol{G}_{11}, \boldsymbol{G}_{12}; \boldsymbol{G}_{21}, \boldsymbol{G}_{22}]$, with

$$\begin{aligned}
\boldsymbol{G}_{11} &= \tfrac{1}{\sigma_c^2}\boldsymbol{H}_1^\mathsf{T}\boldsymbol{H}_1 \boxtimes \boldsymbol{I}_R + \tfrac{1}{\sigma_1^2}\boldsymbol{I}_M \boxtimes \left(\boldsymbol{A}_1^\mathsf{T}\boldsymbol{A}_1 \boxdot \boldsymbol{B}_1^\mathsf{T}\boldsymbol{B}_1\right) \\
\boldsymbol{G}_{22} &= \tfrac{1}{\sigma_c^2}\boldsymbol{H}_2^\mathsf{T}\boldsymbol{H}_2 \boxtimes \boldsymbol{I}_R + \tfrac{1}{\sigma_2^2}\boldsymbol{I}_M \boxtimes \left(\boldsymbol{A}_2^\mathsf{T}\boldsymbol{A}_2 \boxdot \boldsymbol{B}_2^\mathsf{T}\boldsymbol{B}_2\right) \\
\boldsymbol{G}_{12} &= -\tfrac{1}{\sigma_c^2}\boldsymbol{H}_1^\mathsf{T}\boldsymbol{H}_2 \boxtimes \boldsymbol{I}_R \\
\boldsymbol{G}_{21} &= -\tfrac{1}{\sigma_c^2}\boldsymbol{H}_2^\mathsf{T}\boldsymbol{H}_1 \boxtimes \boldsymbol{I}_R
\end{aligned} \tag{4.43}$$

  Normalize columns of $\boldsymbol{C}_1$ using the ℓ_2 norm.
end while
OUTPUT: estimated factors $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{C}_i$.

Algorithm 17 Full Alternating LS algorithm for solving the exact coupled CPD problem

INPUT: $\boldsymbol{\mathcal{T}}_i$, σ_c, σ_i, initial factors $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{C}_i$, coupling matrices $\boldsymbol{H}_i$.
Apply two uncoupled ALS to obtain $\boldsymbol{A}_1$, $\boldsymbol{B}_1$, $\boldsymbol{C}_1$, $\boldsymbol{A}_2$, $\boldsymbol{B}_2$ and $\boldsymbol{C}_2$.
Permute initialization factors columns to have minimal $\|\boldsymbol{H}_1\boldsymbol{C}_1 - \boldsymbol{H}_2\boldsymbol{C}_2\|_F^2$.
while convergence criterion is not met **do**
  for i=1:2 **do**
    $\boldsymbol{A}_i = \boldsymbol{T}_{i,(1)} \left(\boldsymbol{B}_i \odot \boldsymbol{C}_i\right) \left(\boldsymbol{B}_i^\mathsf{T}\boldsymbol{B}_i \boxdot \boldsymbol{C}_i^\mathsf{T}\boldsymbol{C}_i\right)^{-1}$
    $\boldsymbol{B}_i = \boldsymbol{T}_{i,(2)} \left(\boldsymbol{A}_i \odot \boldsymbol{C}\right) \left(\boldsymbol{A}_i^\mathsf{T}\boldsymbol{A}_i \boxdot \boldsymbol{C}^\mathsf{T}\boldsymbol{C}\right)^{-1}$
  end for
  Update $\boldsymbol{C}_1$ by solving the following Sylvester equation:

$$\frac{1}{\sigma_c^2}\boldsymbol{H}_1^\mathsf{T}\boldsymbol{H}_1\boldsymbol{C}_1 + \frac{1}{\sigma_1^2}\boldsymbol{C}_1\left(\boldsymbol{A}_1^\mathsf{T}\boldsymbol{A}_1 \boxdot \boldsymbol{B}_1^\mathsf{T}\boldsymbol{B}_1\right) = \frac{1}{\sigma_c^2}\boldsymbol{H}_1^\mathsf{T}\boldsymbol{H}_2\boldsymbol{C}_2 + \frac{1}{\sigma_1^2}\boldsymbol{T}_{1,(3)}\left(\boldsymbol{A}_1 \odot \boldsymbol{B}_1\right) \tag{4.44}$$

  Normalize columns of $\boldsymbol{C}_1$ using the ℓ_2 norm.
  Update $\boldsymbol{C}_2$ by solving the following Sylvester equation:

$$\frac{1}{\sigma_c^2}\boldsymbol{H}_2^\mathsf{T}\boldsymbol{H}_2\boldsymbol{C}_2 + \frac{1}{\sigma_2^2}\boldsymbol{C}_2\left(\boldsymbol{A}_2^\mathsf{T}\boldsymbol{A}_2 \boxdot \boldsymbol{B}_2^\mathsf{T}\boldsymbol{B}_2\right) = \frac{1}{\sigma_c^2}\boldsymbol{H}_2^\mathsf{T}\boldsymbol{H}_1\boldsymbol{C}_1 + \frac{1}{\sigma_2^2}\boldsymbol{T}_{2,(3)}\left(\boldsymbol{A}_2 \odot \boldsymbol{B}_2\right) \tag{4.45}$$

end while
OUTPUT: estimated factors $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{C}_i$.

for all couplings

$$\begin{cases} \boldsymbol{C}_1 = \boldsymbol{C}_2 + \boldsymbol{\Gamma}_{\boldsymbol{\lambda}} \\ \boldsymbol{\Gamma}_{\boldsymbol{\lambda}} \sim \mathcal{AN}\left(\mathbf{0}, \frac{\lambda^2}{\sigma_c^2}\boldsymbol{I} \otimes \boldsymbol{I}\right) \end{cases} \tag{4.47}$$

where $\lambda \in \mathbb{R}^+$. Indeed,

$$\begin{aligned} &\sum_{i=1}^{2} \frac{1}{\sigma_i^2} \left\| \boldsymbol{\mathcal{T}}_i - (\boldsymbol{A}_i \otimes \boldsymbol{B}_i \otimes \boldsymbol{C}_i) \boldsymbol{\mathcal{I}}_{R_i} \right\|_F^2 + \frac{1}{\sigma_c^2} \|\boldsymbol{C}_1 - \boldsymbol{C}_2\|_F^2 \\ &= \sum_{i=1}^{2} \frac{1}{\sigma_i^2} \left\| \boldsymbol{\mathcal{T}}_i - \left(\lambda \boldsymbol{A}_i \otimes \boldsymbol{B}_i \otimes \frac{\boldsymbol{C}_i}{\lambda}\right) \boldsymbol{\mathcal{I}}_{R_i} \right\|_F^2 + \frac{\lambda^2}{\sigma_c^2} \left\| \frac{1}{\lambda} (\boldsymbol{C}_1 - \boldsymbol{C}_2) \right\|_F^2 \end{aligned} . \tag{4.48}$$

As a consequence, if no normalization is imposed on $\boldsymbol{C}_i$ directly in the tensor decomposition model, whatever the coupling noise level is in reality, the algorithm cannot account for it, and any control on the similarity between the two factors imposed by σ_c is lost. Thus in the hybrid coupling model, the coupling relationship has to be written for a particular norm of one of the two factors, for instance:

$$\begin{cases} \boldsymbol{C}_1 = \boldsymbol{C}_2 + \boldsymbol{\Gamma} \\ \|\boldsymbol{c}_r^{(1)}\|_2^2 = 1 \text{ for all } r \\ \boldsymbol{\Gamma} \sim \mathcal{AN}\left(\mathbf{0}, \frac{1}{\sigma_c^2}\boldsymbol{I} \otimes \boldsymbol{I}\right) \end{cases} . \tag{4.49}$$

The norm constraint on $\boldsymbol{C}_i$ can be applied by a simple projection as in Algorithms 16 and 17.

Another minor issue with normalization in hybrid coupled models is that the usual normalization of two out of three factors is too constrained to simply remove the scaling ambiguity. Indeed, since $\boldsymbol{C}_i$ are linked, the norm of columns of $\boldsymbol{C}_1$ fixes the norm of columns of $\boldsymbol{C}_2$. Consequently in ALS algorithms, normalization can be applied for example on $\boldsymbol{A}_1$, $\boldsymbol{C}_1$ and $\boldsymbol{A}_2$ to remove the scaling indeterminacy in both CPD.

4.2.5 Bayesian and hybrid Cramér-Rao bounds

Introduction

A lower bound on the MSE can be evaluated using variations of the Cramér-Rao bound (CRB). CRBs for the simple CPD are developed in Appendix B. In what follows we will consider the CRB for two simplifications of the joint Gaussian model, namely the simplified coupling model and unitary first rows for uncoupled factors.

Simplified coupling model and unitary first rows in both models we consider Gaussian couplings between components $\boldsymbol{c}_r^{(1)}$ and $\boldsymbol{c}_r^{(2)}$ of factors $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ of the form

$$\boldsymbol{c}_r^{(1)} = \boldsymbol{H}\boldsymbol{c}_r^{(2)} + \boldsymbol{\gamma}_r, \quad \text{for } r \in \{1, , \cdots, R\} \tag{4.50}$$

where $\boldsymbol{H}$ is a transformation matrix and $\boldsymbol{\gamma}_r$ is a zero mean white Gaussian vector with variance σ_c^2. Similarly to [84, 111], to resolve the scale ambiguity of the CP models, we assume that the components in factors $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{A}'$ and $\boldsymbol{B}'$ are of the form $\boldsymbol{a} = [1; \tilde{\boldsymbol{a}}]$, that is, the elements in their first row are known and equal to unity. Without proper constraints on the factors, the information matrices that will be defined later are not invertible and, as a consequence, the bound cannot be evaluated.

Priors on the factors we consider a Bayesian model and a hybrid model. They differ in the definition of the priors of all factors except $\boldsymbol{C}_2$. In the Bayesian model the unknown parts of the components of $\boldsymbol{A}_1$, $\boldsymbol{B}_1$, $\boldsymbol{A}_2$ and $\boldsymbol{B}_2$ have Gaussian priors of the form

$$\tilde{\boldsymbol{a}}_r = \bar{\boldsymbol{a}}_r + \boldsymbol{\gamma}_{\boldsymbol{a}_r} \tag{4.51}$$

where $\bar{\boldsymbol{a}}_r$ is a constant vector representing the mean of the prior and $\boldsymbol{\gamma}_{\boldsymbol{a}_r}$ is a zero mean white Gaussian vector with variance $\sigma_{\boldsymbol{A}}^2$. The components of the factor $\boldsymbol{C}_2$ have a similar prior, except that the elements in the first row are also unknown:

$$\boldsymbol{c}_r^{(2)} \sim \mathcal{N}\left(\bar{\boldsymbol{c}}_r^{(2)}, \sigma_2^2 \boldsymbol{I}\right). \tag{4.52}$$

Sub matrices	$\boldsymbol{F} \times \sigma_n^2$	$\mathbb{E}\{\boldsymbol{F}\} \times \sigma_n^2$
$(\tilde{\boldsymbol{a}}_r, \tilde{\boldsymbol{a}}_s)$	$(\tilde{\boldsymbol{b}}_r^\mathsf{T}\tilde{\boldsymbol{b}}_s + 1)(\boldsymbol{c}_r^\mathsf{T}\boldsymbol{c}_s)\boldsymbol{I}$	$[\bar{\boldsymbol{b}}_r^\mathsf{T}\bar{\boldsymbol{b}}_s + 1 + (J-1)\sigma_{\boldsymbol{B}}^2\delta_{rs}]\left\{\bar{\boldsymbol{c}}_r^\mathsf{T}\boldsymbol{H}^\mathsf{T}\boldsymbol{H}\bar{\boldsymbol{c}}_s + [K\sigma_c^2 + \mathrm{tr}(\boldsymbol{H}^\mathsf{T}\boldsymbol{H})\sigma_{\boldsymbol{C}_2}^2]\right\}\boldsymbol{I}$
$(\tilde{\boldsymbol{a}}_r, \tilde{\boldsymbol{b}}_s)$	$(\boldsymbol{c}_r^\mathsf{T}\boldsymbol{c}_s)(\tilde{\boldsymbol{a}}_s \boxtimes \tilde{\boldsymbol{b}}_r^\mathsf{T} + \boldsymbol{I})$	$\left\{\bar{\boldsymbol{c}}_r^\mathsf{T}\boldsymbol{H}^\mathsf{T}\boldsymbol{H}\bar{\boldsymbol{c}}_s + [K\sigma_c^2 + \mathrm{tr}(\boldsymbol{H}^\mathsf{T}\boldsymbol{H})\sigma_{\boldsymbol{C}_2}^2]\right\}(\bar{\boldsymbol{a}}_s \boxtimes \bar{\boldsymbol{b}}_r^\mathsf{T} + \boldsymbol{I})$
$(\tilde{\boldsymbol{a}}_r, \boldsymbol{c}_s)$	$(\tilde{\boldsymbol{b}}_r^\mathsf{T}\tilde{\boldsymbol{b}}_s + 1)(\tilde{\boldsymbol{a}}_s \boxtimes \boldsymbol{c}_r^\mathsf{T})$	$[\bar{\boldsymbol{b}}_r^\mathsf{T}\bar{\boldsymbol{b}}_s + 1 + (J-1)\sigma_{\boldsymbol{B}}^2\delta_{rs}](\bar{\boldsymbol{a}}_s \boxtimes \boldsymbol{H}(\bar{\boldsymbol{c}}_r^{(2)})^\mathsf{T})$
$(\tilde{\boldsymbol{b}}_r, \tilde{\boldsymbol{b}}_s)$	$(\tilde{\boldsymbol{a}}_r^\mathsf{T}\tilde{\boldsymbol{a}}_s + 1)(\boldsymbol{c}_r^\mathsf{T}\boldsymbol{c}_s)\boldsymbol{I}$	$(\bar{\boldsymbol{a}}_r^\mathsf{T}\bar{\boldsymbol{a}}_s + 1 + I\sigma_{\boldsymbol{A}}^2\delta_{rs})\left\{\bar{\boldsymbol{c}}_r^\mathsf{T}\boldsymbol{H}^\mathsf{T}\boldsymbol{H}\bar{\boldsymbol{c}}_s + [K\sigma_c^2 + \mathrm{tr}(\boldsymbol{H}^\mathsf{T}\boldsymbol{H})\sigma_{\boldsymbol{C}_2}^2]\right\}\boldsymbol{I}$
$(\tilde{\boldsymbol{b}}_r, \boldsymbol{c}_s)$	$(\tilde{\boldsymbol{a}}_r^\mathsf{T}\tilde{\boldsymbol{a}}_s + 1)(\tilde{\boldsymbol{b}}_s \boxtimes \boldsymbol{c}_r^\mathsf{T})$	$[\bar{\boldsymbol{a}}_r^\mathsf{T}\bar{\boldsymbol{a}}_s + 1 + (I-1)\sigma_{\boldsymbol{A}}^2\delta_{rs}](\bar{\boldsymbol{b}}_s \boxtimes \boldsymbol{H}(\bar{\boldsymbol{c}}_r^{(2)})^\mathsf{T})$
$(\boldsymbol{c}_r, \boldsymbol{c}_s)$	$(\tilde{\boldsymbol{a}}_r^\mathsf{T}\tilde{\boldsymbol{a}}_s + 1)(\tilde{\boldsymbol{b}}_r^\mathsf{T}\tilde{\boldsymbol{b}}_s + 1)\boldsymbol{I}$	$[\bar{\boldsymbol{a}}_r^\mathsf{T}\bar{\boldsymbol{a}}_s + 1 + (I-1)\sigma_{\boldsymbol{A}}^2\delta_{rs}][\bar{\boldsymbol{b}}_r^\mathsf{T}\bar{\boldsymbol{b}}_s + 1 + (J-1)\sigma_{\boldsymbol{B}}^2\delta_{rs}]\boldsymbol{I}$

Table 4.1: Fisher information and average Fisher information sub matrices for pairs of components. r is the row block index while s is the column block index. The (1) top index is dropped on parameters related to the first tensor to clarify the computations.

In these conditions, there cannot be an additional prior on $\boldsymbol{C}_1$ since $p(\boldsymbol{C}_1) = p(\boldsymbol{C}_1|\boldsymbol{C}_2)p(\boldsymbol{C}_2)$ by the Bayes rule, and both $p(\boldsymbol{C}_1|\boldsymbol{C}_2)$ and $p(\boldsymbol{C}_2)$ have already been provided.

In the hybrid model, we consider that only $\boldsymbol{C}_1$ is random, all the other factors are deterministic. Then only the conditional probability $p(\boldsymbol{C}_1|\boldsymbol{C}_2)$ is defined.

Likelihoods the noise follows the model in (4.26), so that the log-likelihoods can be written as a function of the components in the factors as follows

$$\begin{aligned}\mathcal{L}_1 &= \log p(\boldsymbol{\mathcal{Y}}_1; \tilde{\boldsymbol{A}}_1, \tilde{\boldsymbol{B}}_1, \boldsymbol{C}_1) = \\ &-\frac{1}{\sigma_1^2}\left(\|\boldsymbol{y}_{\tilde{\boldsymbol{a}}\tilde{\boldsymbol{b}}\boldsymbol{c}}^{(1)} - \sum_{r=1}^{R}\tilde{\boldsymbol{a}}_r^{(1)} \boxtimes \tilde{\boldsymbol{b}}_r^{(1)} \boxtimes \boldsymbol{c}_r^{(1)}\|^2 + \|\boldsymbol{y}_{\tilde{\boldsymbol{a}}\boldsymbol{c}}^{(1)} - \sum_{r=1}^{R}\tilde{\boldsymbol{a}}_r^{(1)} \boxtimes \boldsymbol{c}_r^{(1)}\|^2\right. \\ &\left.+\|\boldsymbol{y}_{\tilde{\boldsymbol{b}}\boldsymbol{c}}^{(1)} - \sum_{r=1}^{R}\tilde{\boldsymbol{b}}_r^{(1)} \boxtimes \boldsymbol{c}_r^{(1)}\|^2 + \|\boldsymbol{y}_{\boldsymbol{c}}^{(1)} - \sum_{r=1}^{R}\boldsymbol{c}_r^{(1)}\|^2\right)\end{aligned} \tag{4.53}$$

where $\boldsymbol{y}$ indicates the vectorized version of the data array as described in section 1.1.1 and the subscripts indicate the free variables underlying the elements in the measurement vector. For example, in $\boldsymbol{y}_{\tilde{\boldsymbol{a}}\boldsymbol{c}}^{(1)}$ we have coordinates from the first tensor which do not contain index 1 in the first mode and contain only index 1 in the second mode.

Bayesian Cramér-Rao bound

When all factors are random, we can obtain a bound on the estimation MSE through the Bayesian Cramér-Rao bound (BCRB) [115, pp.4–5]. For a vector of parameters

$$\tilde{\theta} = \begin{bmatrix} \mathrm{vec}\left(\tilde{\boldsymbol{A}}_1\right) \\ \mathrm{vec}\left(\tilde{\boldsymbol{B}}_1\right) \\ \mathrm{vec}\left(\boldsymbol{C}_1\right) \\ \mathrm{vec}\left(\tilde{\boldsymbol{A}}_2\right) \\ \mathrm{vec}\left(\tilde{\boldsymbol{B}}_2\right) \\ \mathrm{vec}\left(\boldsymbol{C}_2\right) \end{bmatrix} \tag{4.54}$$

we have

$$\mathrm{MSE}(\tilde{\theta}_i) \geqslant \mathrm{BCRB}_{ii} \tag{4.55}$$

where the BCRB is a matrix given by the inverse of a data related information matrix $\boldsymbol{F}$ plus a prior related information matrix $\boldsymbol{P}$:

$$\mathrm{BCRB} = \left(\bar{\boldsymbol{F}} + \boldsymbol{P}\right)^{-1} \tag{4.56}$$

In what follows, we describe both matrices $\bar{\boldsymbol{F}}$ and $\boldsymbol{P}$. $\bar{\boldsymbol{F}}$ is the average Fisher information matrix given by

$$\bar{\boldsymbol{F}} = \begin{bmatrix} \mathbb{E}[\boldsymbol{F}_1] & \boldsymbol{0} \\ \boldsymbol{0} & \mathbb{E}[\boldsymbol{F}_2] \end{bmatrix} \tag{4.57}$$

where the expectation is evaluated w.r.t. the prior distributions. The off-diagonal matrices are equal to zero due to the conditional independence assumption (H1). $\boldsymbol{F}$ is the Fisher information matrix for the first CP model:

$$\boldsymbol{F}_1 = \mathbb{E}\left\{ \begin{bmatrix} \nabla_{\tilde{\boldsymbol{A}}}\mathcal{L}_1 \\ \nabla_{\tilde{\boldsymbol{B}}}\mathcal{L}_1 \\ \nabla_{\boldsymbol{C}}\mathcal{L}_1 \end{bmatrix} \begin{bmatrix} \nabla^{\mathsf{T}}_{\tilde{\boldsymbol{A}}}\mathcal{L}_1 & \nabla^{\mathsf{T}}_{\tilde{\boldsymbol{B}}}\mathcal{L}_1 & \nabla^{\mathsf{T}}_{\boldsymbol{C}}\mathcal{L}_1 \end{bmatrix} \right\} \tag{4.58}$$

where $\nabla_{\boldsymbol{x}}\mathcal{L}_1$ corresponds to the gradient of the log-likelihoods for the first tensor w.r.t. to a vector $\boldsymbol{x}$. The expectation is evaluated w.r.t. the conditional distribution of the data array given the parameters. The matrix $\boldsymbol{F}'$ for the other CP model is given in a similar way. The gradients w.r.t. components $\tilde{\boldsymbol{a}}_r^{(1)}$, $\tilde{\boldsymbol{b}}_r^{(1)}$ and $\boldsymbol{c}_r^{(1)}$, which are parts of the gradients above, are

$$\begin{aligned} \nabla_{\tilde{\boldsymbol{a}}_r}\mathcal{L}_1 &= -\frac{1}{\sigma_1^2}\left[(\boldsymbol{I}\boxtimes\tilde{\boldsymbol{b}}_r^{(1)}\boxtimes\boldsymbol{c}_r^{(1)})^{\mathsf{T}}\boldsymbol{e}^{(1)}_{\tilde{\boldsymbol{a}}\tilde{\boldsymbol{b}}\boldsymbol{c}} + (\boldsymbol{I}\boxtimes\boldsymbol{c}_r^{(1)})^{\mathsf{T}}\boldsymbol{e}^{(1)}_{\tilde{\boldsymbol{a}}\boldsymbol{c}}\right] \\ \nabla_{\tilde{\boldsymbol{b}}_r}\mathcal{L}_1 &= -\frac{1}{\sigma_n^2}\left[(\tilde{\boldsymbol{a}}_r^{(1)}\boxtimes\boldsymbol{I}\boxtimes\boldsymbol{c}_r^{(1)})^{\mathsf{T}}\boldsymbol{e}^{(1)}_{\tilde{\boldsymbol{a}}\tilde{\boldsymbol{b}}\boldsymbol{c}} + (\boldsymbol{I}\boxtimes\boldsymbol{c}_r^{(1)})^{\mathsf{T}}\boldsymbol{e}^{(1)}_{\tilde{\boldsymbol{b}}\boldsymbol{c}}\right] \\ \nabla_{\boldsymbol{c}_r}\mathcal{L}_1 &= -\frac{1}{\sigma_n^2}\left[(\tilde{\boldsymbol{a}}_r^{(1)}\boxtimes\tilde{\boldsymbol{b}}_r^{(1)}\boxtimes\boldsymbol{I})^{\mathsf{T}}\boldsymbol{e}^{(1)}_{\tilde{\boldsymbol{a}}\tilde{\boldsymbol{b}}\boldsymbol{c}} + (\boldsymbol{a}_r^{(1)}\boxtimes\boldsymbol{I})^{\mathsf{T}}\boldsymbol{e}^{(1)}_{\tilde{\boldsymbol{a}}\boldsymbol{c}}\right. \\ &\qquad \left. + (\boldsymbol{b}_r^{(1)}\boxtimes\boldsymbol{I})^{\mathsf{T}}\boldsymbol{e}^{(1)}_{\tilde{\boldsymbol{b}}\boldsymbol{c}} + \boldsymbol{e}^{(1)}_{\boldsymbol{c}}\right] \end{aligned} \tag{4.59}$$

where $\boldsymbol{e}^{(1)}$ indicates a vector of the residuals, *e.g.* $\boldsymbol{e}^{(1)}_{\tilde{\boldsymbol{a}}\boldsymbol{c}} = \boldsymbol{y}^{(1)}_{\tilde{\boldsymbol{a}}\boldsymbol{c}} - \sum_{r=1}^{R}\tilde{\boldsymbol{a}}_r^{(1)}\boxtimes\boldsymbol{c}_r^{(1)}$, which corresponds here to a Gaussian noise vector. Using the fact that the noise is white and that $(\boldsymbol{x}\boxtimes\boldsymbol{y}\boxtimes\boldsymbol{z})^{\mathsf{T}}(\boldsymbol{x}'\boxtimes\boldsymbol{y}'\boxtimes\boldsymbol{z}') = (\boldsymbol{x}^{\mathsf{T}}\boldsymbol{x}')(\boldsymbol{y}^{\mathsf{T}}\boldsymbol{y}')(\boldsymbol{z}^{\mathsf{T}}\boldsymbol{z}')$, we can easily evaluate the submatrices in $\boldsymbol{F}$. They are given in Table 4.1, where δ_{rs} denotes the discrete delta function. Similar expressions are obtained for $\boldsymbol{F}_2$.

Using the coupling and priors models (4.50) and (4.51) and the independence hypothesis on the uncoupled factors (H2), we can obtain the average Fisher information matrix ($\bar{\boldsymbol{F}}$), which is also given in Table 4.1.

The matrix $\boldsymbol{P}$ is the information matrix related to the prior distribution $p(\tilde{\boldsymbol{\theta}})$, it is given by

$$\boldsymbol{P} = \mathbb{E}\left[\nabla_{\tilde{\boldsymbol{\theta}}}\log p(\tilde{\boldsymbol{\theta}})\nabla^{\mathsf{T}}_{\tilde{\boldsymbol{\theta}}}\log p(\tilde{\boldsymbol{\theta}})\right]. \tag{4.60}$$

Using again the prior models and (H2), we have

$$\boldsymbol{P} = \begin{bmatrix} \frac{\boldsymbol{I}}{\sigma^2_{\boldsymbol{A}_1}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \frac{\boldsymbol{I}}{\sigma^2_{\boldsymbol{B}_1}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \frac{\boldsymbol{I}}{\sigma^2_{\boldsymbol{c}}} & \boldsymbol{0} & \boldsymbol{0} & -\frac{\boldsymbol{I}\boxtimes\boldsymbol{H}}{\sigma^2_c} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \frac{\boldsymbol{I}}{\sigma^2_{\boldsymbol{A}_2}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \frac{\boldsymbol{I}}{\sigma^2_{\boldsymbol{B}_2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & -\frac{\boldsymbol{I}\boxtimes\boldsymbol{H}^{\mathsf{T}}}{\sigma^2_c} & \boldsymbol{0} & \boldsymbol{0} & \frac{\boldsymbol{I}\boxtimes(\boldsymbol{H}^{\mathsf{T}}\boldsymbol{H})}{\sigma^2_c} + \frac{\boldsymbol{I}}{\sigma^2_{\boldsymbol{C}_2}} \end{bmatrix}. \tag{4.61}$$

The off-diagonal sub matrices are the effects of coupling between the two CP models on the information matrix to be inverted, the larger is σ_c^2, the smaller is the influence of these sub matrices.

Hybrid Cramér-Rao bound

If we consider the hybrid model where only $\boldsymbol{C}_1$ is random, a bound on the estimation MSE can be obtained through the hybrid Cramér-Rao bound (HCRB) ([115, p. 12]). The HCRB is given by

$$\text{HCRB} = \left(\bar{\boldsymbol{F}}^{\boldsymbol{C}_1|\boldsymbol{C}_2} + \boldsymbol{P}^{\boldsymbol{C}_1|\boldsymbol{C}_2}\right)^{-1}. \tag{4.62}$$

Matrices $\bar{\boldsymbol{F}}^{\boldsymbol{C}_1|\boldsymbol{C}_2}$ and $\boldsymbol{P}^{\boldsymbol{C}_1|\boldsymbol{C}_2}$ play similar roles as $\bar{\boldsymbol{F}}$ and $\boldsymbol{P}$ for the BCRB. The matrix $\bar{\boldsymbol{F}}^{\boldsymbol{C}_1|\boldsymbol{C}_2}$ is evaluated in a similar way as $\bar{\boldsymbol{F}}$ with the difference that the expectation is taken w.r.t. the conditional distribution of the coupled factor $p(\boldsymbol{C}_1|\boldsymbol{C}_2)$ only, that is

$$\bar{\boldsymbol{F}}^{\boldsymbol{C}_1|\boldsymbol{C}_2} = \begin{bmatrix} \mathbb{E}_{\boldsymbol{C}_1|\boldsymbol{C}_2}[\boldsymbol{F}_1] & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{F}_2 \end{bmatrix}. \tag{4.63}$$

The submatrix $\boldsymbol{F}_2$ is directly the Fisher information for the second CP model and the elements in $\mathbb{E}_{\boldsymbol{C}_1|\boldsymbol{C}_2}[\boldsymbol{F}_1]$ are similar to the elements in $\boldsymbol{F}$, except that only the components $\boldsymbol{c}_r^{(1)}$ are affected by the expectation. For the hybrid model $\mathbb{E}[\boldsymbol{c}_r^{(1)}] = \boldsymbol{c}_r^{(2)}$ and $\mathbb{E}[(\boldsymbol{c}_r^{(1)})^\mathsf{T}\boldsymbol{c}_s^{(1)}] = (\boldsymbol{c}_r^{(2)})^\mathsf{T}\boldsymbol{H}^\mathsf{T}\boldsymbol{H}\boldsymbol{c}_s^{(1)} + K\sigma_c^2\delta_{rs}$.

The prior matrix is evaluated w.r.t. the distribution of the random parameters $\tilde{\boldsymbol{\theta}}^r$ conditioned on the non random parameters $\tilde{\boldsymbol{\theta}}^{nr}$

$$\boldsymbol{P}^{\boldsymbol{C}_1|\boldsymbol{C}_2} = \mathbb{E}_{\tilde{\boldsymbol{\theta}}^r|\tilde{\boldsymbol{\theta}}^{nr}}\left[\nabla_{\tilde{\boldsymbol{\theta}}}\log p(\tilde{\boldsymbol{\theta}}^r|\tilde{\boldsymbol{\theta}}^{nr})\nabla_{\tilde{\boldsymbol{\theta}}}^\mathsf{T}\log p(\tilde{\boldsymbol{\theta}}^r|\tilde{\boldsymbol{\theta}}^{nr})\right]. \tag{4.64}$$

For the hybrid coupled CP model we have

$$\boldsymbol{P} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\boldsymbol{I}}{\sigma_c^2} & \mathbf{0} & \mathbf{0} & -\frac{\boldsymbol{I}\boxtimes\boldsymbol{H}}{\sigma_c^2} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\frac{\boldsymbol{I}\boxtimes\boldsymbol{H}^\mathsf{T}}{\sigma_c^2} & \mathbf{0} & \mathbf{0} & \frac{\boldsymbol{I}\boxtimes(\boldsymbol{H}^\mathsf{T}\boldsymbol{H})}{\sigma_c^2} \end{bmatrix}. \tag{4.65}$$

We can still see the off diagonal sub matrices indicating the coupling, but parts of the diagonal elements are now equal to zero, which means that the hybrid model is less regularized than the Bayesian model.

4.2.6 Application to joint decomposition of fluorescence and nuclear magnetic resonance data

Let us illustrate the advantages of flexible coupling models with respect to the exact coupling model and two uncoupled CPD on the coupled fluorescence/NMR data set. We aim to show that toying with the coupling intensity σ_c means controlling the amount of similarity imposed on the coupled factors.

Recall that in the coupled spectroscopy (EEM)/nuclear magnetic resonance (NMR) data set introduced in chapter 2, only some components are observed by both measurement modalities. This means that a coupling constraint is to be imposed only on a part of the columns of $\boldsymbol{C}_{EEM}$ and $\boldsymbol{C}_{NMR}$. How to determine approximately the number of shared components is explained below in section 4.3.2. Moreoever, the components to be extracted are in theory non-negative, and therefore non-negativity constraints are additionnaly imposed in the fusion algorithm 16 through projection on the non-negative orthant.

It is reported by Acar *et. al.*[6] that only three components are visible through fluorescence spectroscopy, and that five chemicals among the five involved in the mixture are visible to NMR. So in theory the rank R_{EEM} is 3, R_{NMR} is 5 and the number of shared components is 3.

Figures 4.4, 4.5 and 4.6 below compare results obtained from running two independant PROCO-ALS, CMTF from Acar et al. (Figure 4.4), exact coupling of the shared components computed through Algorithm 12 (slighly modified to include non-negativity and share only a few components), and the following flexible coupling model computed through Algorithm 16:

$$\begin{cases} \boldsymbol{\mathcal{T}}_{EEM} = (\boldsymbol{A}_{EEM}\otimes\boldsymbol{B}_{EEM}\boldsymbol{C}_{EEM})\boldsymbol{\mathcal{I}}_3 + \boldsymbol{\mathcal{E}}_{EEM} \\ \boldsymbol{\mathcal{T}}_{NMR} = (\boldsymbol{A}_{NMR}\otimes\boldsymbol{B}_{NMR}\boldsymbol{C}_{NMR})\boldsymbol{\mathcal{I}}_5 + \boldsymbol{\mathcal{E}}_{NMR} \\ \boldsymbol{C}_{EEM} = \boldsymbol{C}_{NMR}(r=1,2,3) + \boldsymbol{\Gamma}_c \\ \|\boldsymbol{c}_i^{EEM}\|_1 = 1 \ \forall i \leqslant 3 \\ \boldsymbol{\mathcal{E}}_{EEM} \sim \mathcal{AN}(\mathbf{0}, \boldsymbol{I}_{21}\otimes\boldsymbol{I}_{251}\otimes\boldsymbol{I}_{28}) \\ \boldsymbol{\mathcal{E}}_{NMR} \sim \mathcal{AN}\left(\mathbf{0}, \frac{1}{\sigma_{NMR}^2}\boldsymbol{I}_8\otimes\boldsymbol{I}_{13324}\otimes\boldsymbol{I}_{28}\right) \\ \boldsymbol{\Gamma}_c \sim \mathcal{AN}\left(\mathbf{0}, \frac{1}{\sigma_c^2}\boldsymbol{I}_{28}\otimes\boldsymbol{I}_3\right) \end{cases} \tag{4.66}$$

where σ_c is the coupling intensity, which is unknown (it may not even have a physical interpretation) and will be considered a nuisance parameter standing for flexibility in the equality constraints of the coupling model. The measurement noise variance on the fluorescence data is fixed to $\sigma_{EEM} = 1$ since the MAP estimator is invariant to a positive scaling. The diagonal tensors are set to identity

[6] see `http://www.models.life.ku.dk/joda/prototype`

since the norm of components will be pulled in $\boldsymbol{A}$. To balance the two data fitting terms in the cost function, σ_{NMR} is set to 0.1.

The different models featuring the various coupling intensities are run 20 times, and only the run that exhibits minimal final cost function value is kept for each algorithm. To speed up the computation, joint compression was applied as described in the next section, and used to compress the two coupled data set for all four algorithms. Again the non-negativity constraints are applied using the pseudo-projection developed for PROCO-ALS described in section 3.2.

Missing data are handled in a very suboptimal manner, by replacing unknown values before the data are normalized by realizations of the absolute value of a Gaussian random variable of variance 0.01^2 and zero mean. Therefore these new artificial values are much smaller than the average value in the EEM and NMR data tensors (respectively 2.2×10^4 and 2.7×10^5).

Algorithms run until the maximal number of iterations is reached, which is probably not the best choice of stopping criterion in practice but is not an issue here since we are not interested in computation time. To ensure conergence is reached, the number of iterations is chosen large. Number of iterations for the PROCO-ALS is 3000, but is only set to 500 for the joint algorithms since the results of the two PROCO-ALS are used to initialize the joint algorithms. Estimated factors are normalized columnwise using the ℓ_2 norm after they have been estimated.

The results go as follow: no significant change is observable on the spectra $\boldsymbol{A}$ and $\boldsymbol{B}$ with respect to the coupling model and the value of σ_c, probably because the data are very simple. Indeed, the 5 first measured mixtures contain only one component, which means even a rank one NMF could be used on the first measurements. On the other hand, some differences can be observed between the recovered concentration factors $\boldsymbol{C}$. Clearly the coupling noise level allows to tune the strength of the coupling constraint. Observe how increasing the coupling intensity reduces the amplitude of the differences between columns of C_{EEM} and C_{NMR}. However for this artifical data set where the coupling should be exact, noisy coupling only deteriorates the results with respect to exact an coupled decomposition.

Even though in this simple data set the (noisy) flexible coupling does not prove significantly useful, it should be tested further on more complex data sets, with potentially non-Gaussian noise. On the other hand, flexible coupling include linearly transformed couplings, which have been extensively used in the data fusion literature. This whole section therefore successfully introduces a theoretical framework which includes both noisy couplings and transformed couplings. In what follows, compression is applied to coupled models where the coupling happen in a linearly transformed domain.

4.3 Joint dimensionality reduction

A pitfall of tensor decomposition models accounting for subject variability is that their decomposition algorithms often have greater complexity than their uncoupled counterpart. For instance, the step to estimate $\boldsymbol{C}^*$ jointly in Algorithm 16 involves solving a large linear system. If the number of columns of $\boldsymbol{H}_i$ is of the same order of magnitude than the third dimension M, then this step of Algorithm 16 implies to solve a $2MR \times 2MR$ linear system, which is larger than the usual $R \times R$ linear system for ALS. Moreover the (costly) computation of all Kronecker products is mandatory, which also heavily slows down the coupled algorithm.

Even if fast algorithms were to be developed, there is always a need for speeding up tensor decomposition algorithms, since they are typically applied for various choices of parameters, constraints type and so on. In the previous chapter, tensors are compressed using HOSVD, which allows a huge speed up in decomposition time at the cost of an initial costly compression. Since this compression is only computed once, there is obviously only a gain if decomposition algorithms are to be run several times. However for coupled models, compression had been very little studied. Some early ideas on the compression of coupled data sets are introduced in [45]. In this section, compression for coupled tensor decomposition models is studied. The section is based on a recently submitted paper [33].

4.3.1 Exact coupling

Suppose that two data sets are acquired in the form of tensors $\boldsymbol{\mathcal{T}}_1$ and $\boldsymbol{\mathcal{T}}_2$, and that these data sets are coupled through the exact coupling model (4.6):

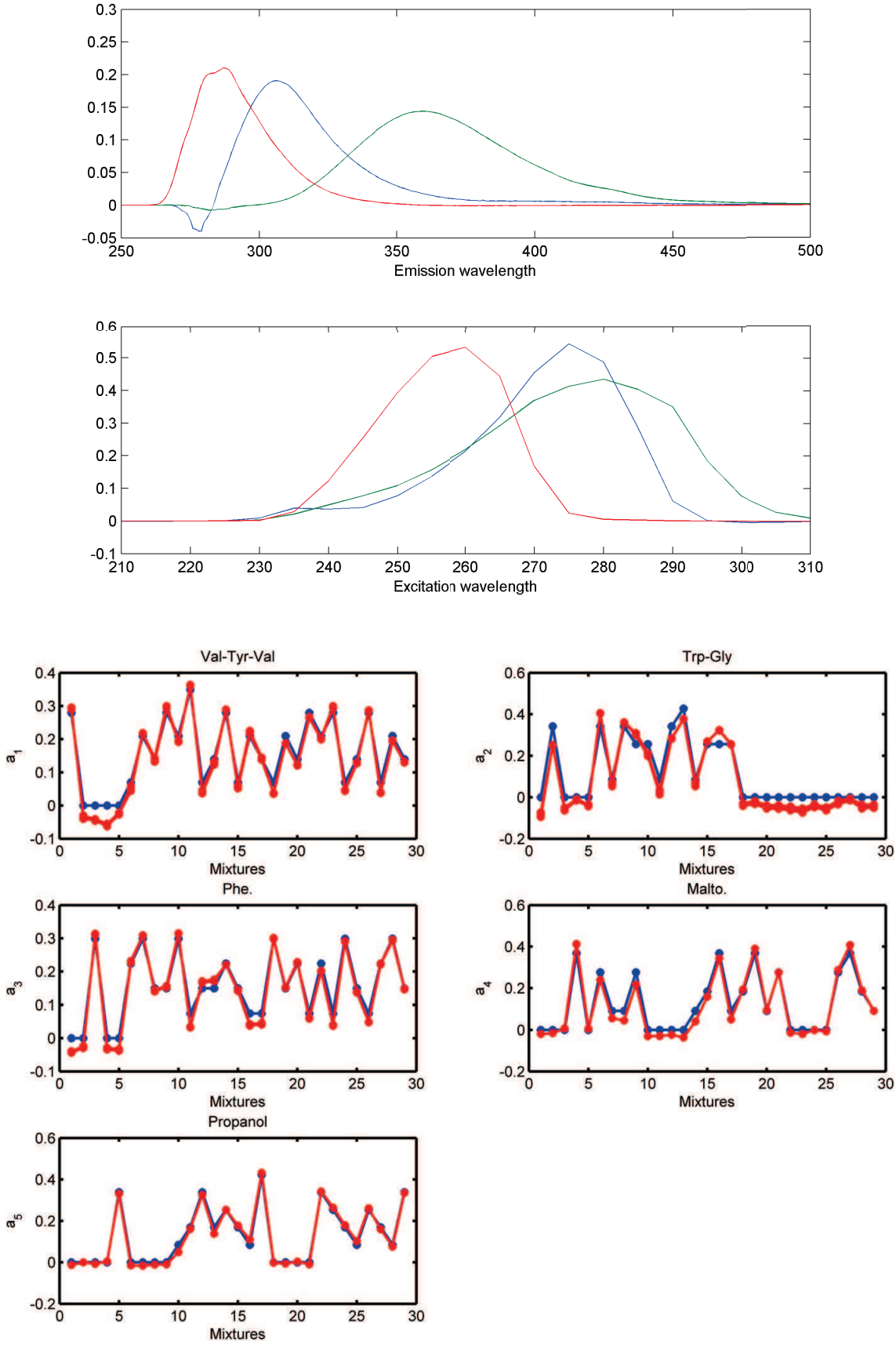
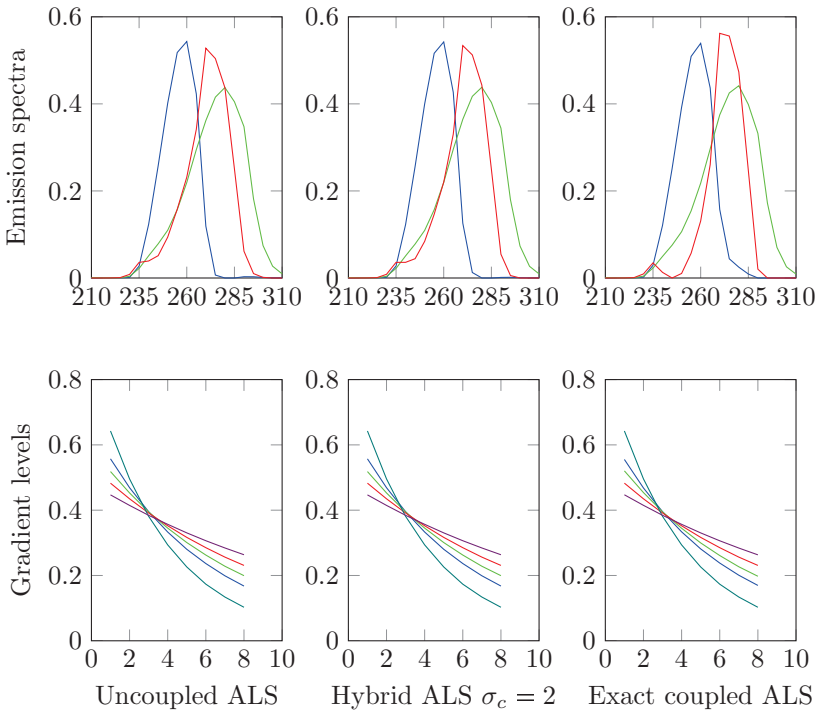
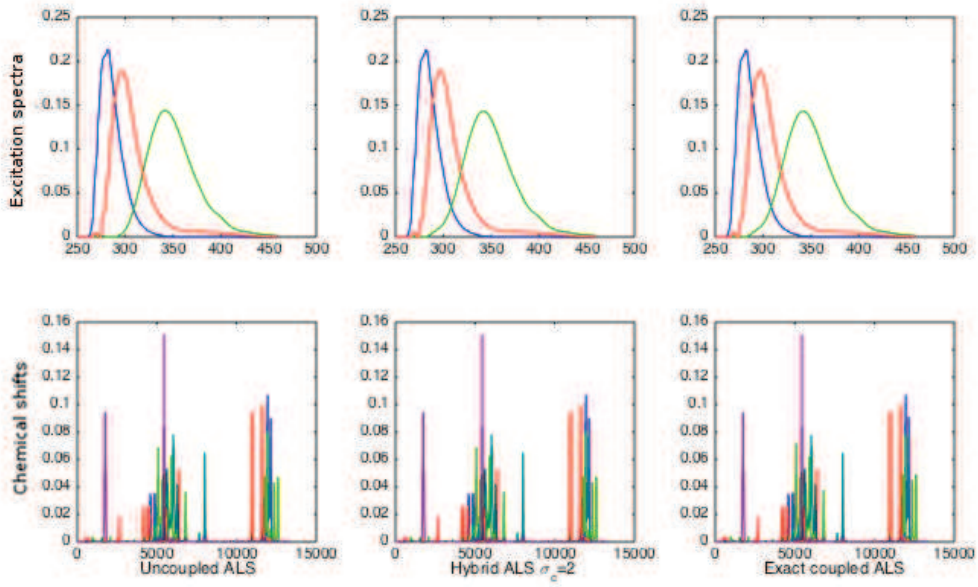


Figure 4.4: Estimated spectra and relative concentrations using CMTF in red (courtesy of Evrim Acar), which handles the partial coupling in an unsupervised manner and computes the CPD using a gradient-based algorithm. These factors were estimated using fluorescence spectroscopy data and NMR data, but not the LC-MS data. In blue are plotted the true relative concentrations.



(a) Top: estimated emission, Bottom: gradient levels



(b) Top: estimated excitation spectra, Bottom: chemical shift

Figure 4.5: Recovered factors. From left to right, uncoupled ALS, flexible ALS (Algorithm 16) and exact coupled ALS. Ranks are 3 and 5, and $\sigma_c = 2$.

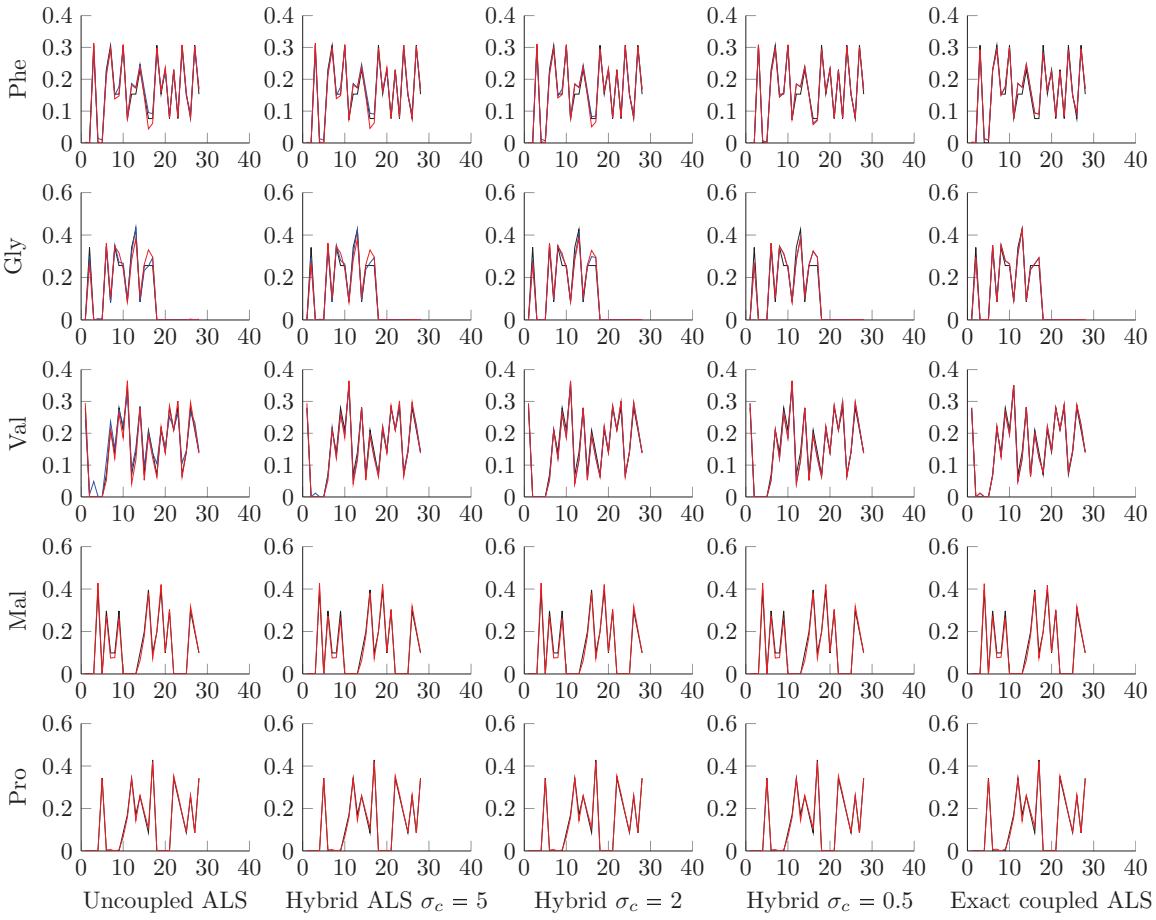


Figure 4.6: Ranks $R_{EEM} = 3$ and $R_{NMR} = 5$ concentration profiles. From left to right: uncoupled ALS, Hybrid ALS (Algorithm 16) with $\sigma_c = 5$, flexible ALS with $\sigma_c = 2$, flexible ALS with $\sigma_c = 0.5$ and exact coupled ALS. In black: true concentration profiles. In blue: components extracted from the EEM. In red: components extracted from the NMR.

$$\begin{cases} \boldsymbol{\mathcal{T}}_1 = (\boldsymbol{A}_1 \otimes \boldsymbol{B}_1 \otimes \boldsymbol{C}_1)\,\boldsymbol{\Sigma}_1 + \boldsymbol{\mathcal{E}}_1 \\ \boldsymbol{\mathcal{T}}_2 = (\boldsymbol{A}_2 \otimes \boldsymbol{B}_2 \otimes \boldsymbol{C}_2)\,\boldsymbol{\Sigma}_2 + \boldsymbol{\mathcal{E}}_2 \\ \boldsymbol{C}_1 = \boldsymbol{C}_2 \end{cases} \tag{4.67}$$

where $\boldsymbol{\mathcal{E}}_1$ and $\boldsymbol{\mathcal{E}}_2$ are the independent measurement noises with variances σ_1^2 and σ_2^2 respectively. The goal is to compress the two data sets. The first question that comes to mind is whether the tensors have to be compressed jointly, or if two independent compressions are satisfactory. As proved below, it turns out joint compression is mandatory to preserve the exact coupling relationship.

The easiest approach is indeed to compress the two coupled tensors independently, and use the information of the coupling only in the CPD computation stage. But there are two issues with this method. First, two independent compressions make no use of the fact that a constraint links the factors. Thus in theory, it cannot be better than a joint compression scheme in terms of compression loss. Indeed, suppose $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are two bases for the third mode of $\boldsymbol{\mathcal{T}}_1$ and $\boldsymbol{\mathcal{T}}_2$. Then exact coupling on the third mode means that $\text{Span}\{\boldsymbol{W}_1\} = \text{Span}\{\boldsymbol{W}_2\}$. Thus only one basis $\boldsymbol{W}_j$ is needed to compress the two tensors on the third mode.

Second, since data are noisy, compression bases would be estimated with some error. However, the coupling relationship is written for the true factors, so that compressing independently would destroy the coupling relationship in the compressed space. Indeed, defining $\boldsymbol{W}_i\boldsymbol{C}_{c_i} = \boldsymbol{C}_i$ for noiseless compression and $\hat{\boldsymbol{W}}_i^T\boldsymbol{C}_i = \hat{\boldsymbol{C}}_{c_i}$ for noisy compression,

$$\boldsymbol{W}_1\boldsymbol{C}_{c_1} = \boldsymbol{W}_2\boldsymbol{C}_{c_2} \text{ but } \hat{\boldsymbol{W}}_1\hat{\boldsymbol{C}}_{c_1} \neq \hat{\boldsymbol{W}}_2\hat{\boldsymbol{C}}_{c_2}. \tag{4.68}$$

More precisely, let δ_i be the estimation error on basis $\boldsymbol{W}_i$, and η_i be the difference between true compressed factors $\boldsymbol{C}_{c_i}$ and compressed factors defined from approximated basis $\hat{\boldsymbol{C}}_{c_i}$ then:

$$\begin{aligned} \boldsymbol{C}_1 = \boldsymbol{C}_2 \quad &\equiv \quad \boldsymbol{W}_1\boldsymbol{C}_{c_1} = \boldsymbol{W}_2\boldsymbol{C}_{c_2} \\ &\equiv \quad \left(\hat{\boldsymbol{W}}_1 + \delta_1\right)\left(\hat{\boldsymbol{C}}_{c_1} + \eta_1\right) = \left(\hat{\boldsymbol{W}}_2 + \delta_2\right)\left(\hat{\boldsymbol{C}}_{c_2} + \eta_2\right) \ . \end{aligned} \tag{4.69}$$

This means that even small estimation error makes an exact coupling constraint in the compress domain inexact. Indeed, the coupled decomposition model in the compressed domain would be:

$$\begin{cases} \boldsymbol{\mathcal{G}}_1 = (\boldsymbol{A}_{c_1} \otimes \boldsymbol{B}_{c_1} \otimes \boldsymbol{C}_{c_1})\,\boldsymbol{\Sigma}_1 + \boldsymbol{\mathcal{E}}_{c_1} \\ \boldsymbol{\mathcal{G}}_2 = (\boldsymbol{A}_{c_2} \otimes \boldsymbol{B}_{c_2} \otimes \boldsymbol{C}_{c_2})\,\boldsymbol{\Sigma}_2 + \boldsymbol{\mathcal{E}}_{c_2} \\ \boldsymbol{C}_{c_1} = \boldsymbol{C}_{c_2} + \boldsymbol{\Gamma} \end{cases} \tag{4.70}$$

and because of the estimation error on the compression basis, some error term $\boldsymbol{\Gamma}$ has to be added to the coupling model, whose distribution is *a priori* unknown.

On the other hand, trying to find the shared basis $\boldsymbol{W}_j$ offers some advantages. First, because more data are used in the estimation process, the estimation error on $\boldsymbol{W}_j$ may be reduced. Moreover, any estimation error on the joint basis does not affect the coupling model in the compressed space:

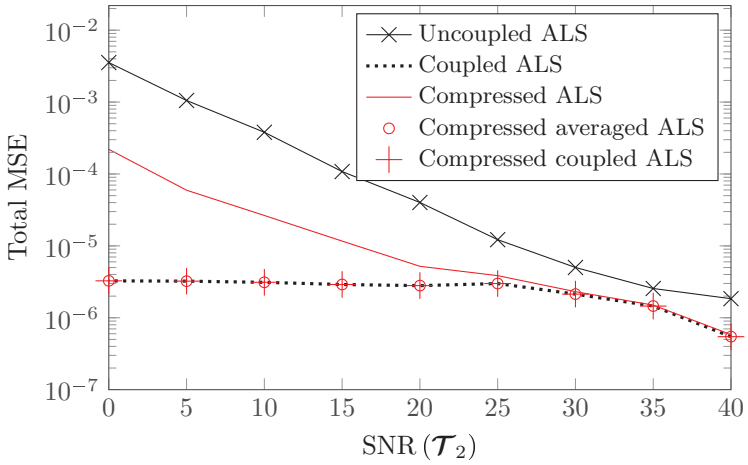
$$\boldsymbol{C}_1 = \boldsymbol{C}_2 \Rightarrow \hat{\boldsymbol{W}}_j^T\boldsymbol{C}_1 = \hat{\boldsymbol{W}}_j^T\boldsymbol{C}_2 \equiv \boldsymbol{C}_{c_1} = \boldsymbol{C}_{c_2} \tag{4.71}$$

where compressed factors are now defined as $\boldsymbol{C}_{c_i} = \hat{\boldsymbol{W}}_j^{\mathsf{T}}\boldsymbol{C}_i$. It is not necessary to consider estimated $\hat{\boldsymbol{C}}_{c_i}$ and true compressed factors $\boldsymbol{C}_{c_i}$, since only the estimated compressed factors are used in the compressed coupled model. Therefore only the notation $\boldsymbol{C}_{c_i}$ is used from now on.

It is clear that the best basis to jointly compress both tensors $\boldsymbol{W}_j$ is obtained by a truncated SVD of the stacked unfolding matrices, as was discussed originally in [45]. If noise levels are different, then they can be taken into account when stacking the unfolding matrices:

$$\left[\frac{\boldsymbol{T}_{1,(3)}}{\sigma_1}\ ,\ \frac{\boldsymbol{T}_{2,(3)}}{\sigma_2}\right] = \hat{\boldsymbol{W}}_j\boldsymbol{\Sigma}_j\boldsymbol{Q}_j. \tag{4.72}$$

Since the dimension of the subspace spanned by the coupled components is R, the SVD can be truncated so that only R singular values and vectors are kept. Moreover by choosing an orthonormal basis, the observation noise is still white in both tensors. Since we have shown that the coupling model is direct even in the compressed space, the compressed optimization problem to solve (4.67) with no noise in the coupling is the following



w

Figure 4.7: Total MSE for estimation of factors $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ for the five described algorithms. MSE is averaged on 100 realizations of the CP model and noise, each estimate is the best among six different initialization points.

$$\begin{array}{ll} \text{minimize} & \frac{1}{\sigma_1^2}\left\|\boldsymbol{\mathcal{G}}_1 - (\boldsymbol{A}_{c_1} \otimes \boldsymbol{B}_{c_1} \otimes \boldsymbol{C}_{c_1})\,\boldsymbol{\Sigma}_1\right\|_F^2 + \frac{1}{\sigma_2^2}\left\|\boldsymbol{\mathcal{G}}_2 - (\boldsymbol{A}_{c_1} \otimes \boldsymbol{B}_{c_2} \otimes \boldsymbol{C}_{c_2})\,\boldsymbol{\Sigma}_2\right\|_F^2 \\ \text{w.r.t.} & \boldsymbol{A}_{c_i}, \boldsymbol{B}_{c_i}, \boldsymbol{C}_{c_i}, \boldsymbol{\Sigma}_i \\ \text{subject to} & \boldsymbol{C}_{c_1} = \boldsymbol{C}_{c_2} \\ \text{where} & \boldsymbol{\mathcal{T}}_1 = (\boldsymbol{U}_1 \otimes \boldsymbol{V}_1 \otimes \boldsymbol{W}_j)\,\boldsymbol{\mathcal{G}}_1 \\ & \boldsymbol{\mathcal{T}}_2 = (\boldsymbol{U}_2 \otimes \boldsymbol{V}_2 \otimes \boldsymbol{W}_j)\,\boldsymbol{\mathcal{G}}_2 \end{array} . \quad (4.73)$$

Some simulations are now run on synthetic coupled data sets to show that coupled compression may not significantly deteriorate estimation of all factors, similarly to what is presented in section 3.2. Three factor matrices of size 50×3 are drawn independently according to a standard Gaussian distribution, but the third factor is exactly the same for both tensors. Then the tensors are normalized to have unit Frobenius norm. Five algorithms to compute the CPD are compared. Three work in a compressed domain of size $3 \times 3 \times 3$. Compression is computed jointly, but then only two algorithms consider coupling in the compressed domain, one being the weighted average ALS described below in section 4.4.1 while the second one is the (compressed) exact coupled ALS Algorithm 12. The third one is an uncoupled ALS for each compressed tensor. Two other algorithms do not compress the initial tensors. One just runs two independent ALS, the other is the exact coupled ALS Algorithm 12 but in the uncompressed domain. The total mean squared error (MSE) on the coupled factors is plotted in Figure 4.7, the SNR of the first tensor is fixed to approximately 33dB ($\sigma_1 = 10^{-4}$) while the other varies from 0 to 40dB.

It appears from the simulation results that the coupling relationship is crucial in the compression. Indeed, the jointly compressed independent ALS algorithm has smaller total MSE than the uncompressed independent ALS. Since compression can also be seen as denoising, it can be concluded that joint data compression helps denoising the noisy data set using information about the span of coupled factors contained in less noisy data. This is not intuitive, since in theory compressing should increase estimation error by reducing the amount of information contained in a data set. Moreover, even without including the coupling knowledge in the ALS algorithm, it is already efficient to simply compress jointly in order to compute small independent ALS. That is why an extension of joint compression to more complex models would be of crucial importance.

Joint compression is also used for computing the coupled models of the previous section 4.2.6, and it can be seen on the results that compressing does not hinder the recovery nor the interpretation of the results.

The next sections introduce extensions of coupled compression to more complex setting, although these are still current research topics with some remaining open problems.

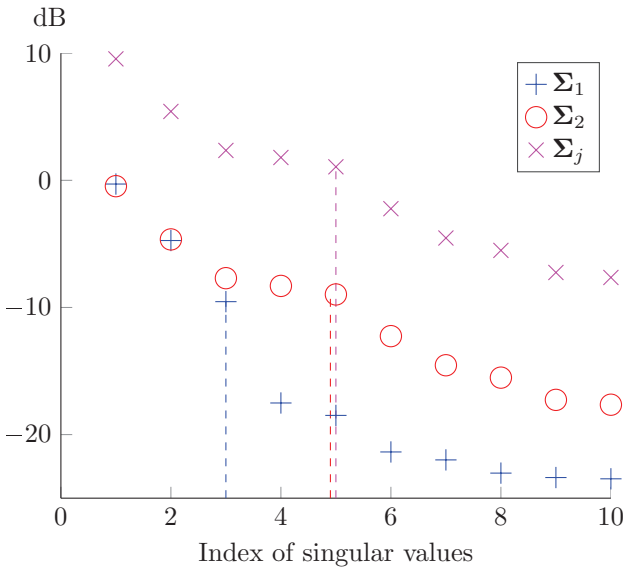


Figure 4.8: Singular values $\boldsymbol{\Sigma}_1$ of $\boldsymbol{T}_{1,(3)}$ in blue, $\boldsymbol{\Sigma}_2$ of $\boldsymbol{T}_{2,(3)}$ and $\boldsymbol{\Sigma}_j$ as in (4.72). Noise levels are $\sigma_1 = 1$ and $\sigma_2 = 0.1$. Vertical lines show where the singular values decrease strongly, suggesting approximate values for the multilinear ranks which are supposed equal to the tensor ranks for this data set. The purple vertical line is related to the number of shared components as described in section 4.3.2.

4.3.2 Shared components

Now if the coupling relationship is still exact but only relates a subset of $r < R$ components, *i.e.* if the data follow a shared components model as described in Example 6 with r coupled components, then compression can be computed in a similar fashion as described above. In this case however, $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ each span two subspaces whose intersection is a subspace of dimension r. Therefore a basis which span contains all columns of both $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ should span three subspaces: the shared subspace of dimension r, and two subspaces of dimensions $R_1 - r$ and $R_2 - r$ that contain uncoupled columns of $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$. In short, when computing the truncated SVD of the stacked weighted matricized arrays as in (4.72), $R_1 + R_2 - r$ singular vectors and values should be kept instead of r.

To compute joint compression when only some components are shared, it is therefore necessary to estimate the number of shared components, or at least to lower bound it. This can be done by studying the singular value profiles of each data set independently, and estimating the number of significant singular values for each data set. They stand for multilinear ranks, which are generically equal to R_1 and R_2 if no collinearity among components is to be found in the data. The number of significant singular values in the joint SVD (4.72) should similarly stand for $R_1 + R_2 - r$. Figure 4.8 shows the singular values profiles for the fluorescence/NMR data set.

4.3.3 Linear transformed coupling

If the coupling is noiseless but expressed through a linear transformation $\boldsymbol{H}$, *i.e.*

$$\boldsymbol{C}_1 = \boldsymbol{H}\boldsymbol{C}_2 \tag{4.74}$$

then a simple multiplication by $\boldsymbol{H}$ of the third mode of the second tensor yields exact coupling:

$$\boldsymbol{\mathcal{Y}}_2 = (\boldsymbol{I} \otimes \boldsymbol{I} \otimes \boldsymbol{H})\,\boldsymbol{\mathcal{T}}_2 = (\boldsymbol{A}_2 \otimes \boldsymbol{B}_2 \otimes \boldsymbol{C}_1)\,\boldsymbol{\Sigma}_2 + \boldsymbol{\mathcal{E}}'_2 \tag{4.75}$$

where $\boldsymbol{\mathcal{E}}'_2$ is correlated on the third mode. That is, $\boldsymbol{\mathcal{E}}'_2$ follows an array normal distribution with Kronecker product covariance $\boldsymbol{I} \boxtimes \boldsymbol{I} \boxtimes \boldsymbol{H}\boldsymbol{H}^T$, which can be handled quite easily in an ALS algorithm as explained in section 1. This means that the exact coupling model can be used in concordance with correlated noise to tackle linear transformations in the coupling, as long as $\boldsymbol{H}\boldsymbol{H}^T$ is invertible. If this is not the case, then the noise distribution after preprocessing is degenerate, *i.e.* does not

admit a density function. Finally, factor $\boldsymbol{C}_2$ can be obtained from the estimated factor $\boldsymbol{C}_1$ by computing the left inverse of $\boldsymbol{H}$. Therefore, an ideal setting for using preprocessing is obtained when the coupling matrices are square and full rank.

The idea of preprocessing the data before running joint decomposition is appealing at first, since in exact transformed coupling scenarios like (4.74), this allows to remove the linear coupling function from the coupling model. Actually, this is exactly what is done in PARAFAC2. Indeed, once the coupling matrices $\boldsymbol{P}_i$ are estimated, the data are preprocessed by multiplying $\boldsymbol{P}_i$ on the third mode of each tensor $\boldsymbol{\mathcal{T}}_i$. But the lingering problem with preprocessing is that non-square coupling matrices yield either non-feasibility of the processed tensor decomposition if the noise distribution is degenerate, or an underdetermined linear system linking the estimated factors and the actual factors, e.g. $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ in (4.74).

Yet, if joint compression is to be applied to exact linearly transformed coupling, to apply the previously developped methodology, it is necessary to first preprocess the data. Indeed before preprocessing, columns of $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ do not span the same subspace, so that a joint basis $\boldsymbol{W}_j$ does not exist. However after preprocessing, since the coupling model becomes exact, joint compression as described in the previous section can be used.

A careful reader will have noticed that the latent factors have been put aside of the discussion in the last pages. Latent factors $\boldsymbol{C}^*$ are again discussed below in section 4.4.1. But at this stage, it is possible to explain why the size of the latent factor $\boldsymbol{C}^*$ in PARAFAC2 does not need to be greater than the rank R. With notations from section 4.1.3, suppose the size of the latent space is some integer M. Clearly M should be greater than R, except if all coupled factors have collinear columns. However if M is strictly larger than R, then the coupling relationships can be written as follows:

$$\boldsymbol{C}_i = \boldsymbol{P}_i\boldsymbol{C}^* \;\equiv \boldsymbol{C}_i = \boldsymbol{P}_i\boldsymbol{W}_j\boldsymbol{C}_c^* \;\equiv\; \boldsymbol{C}_i = \boldsymbol{Q}_i\boldsymbol{C}_c^* \tag{4.76}$$

where $\boldsymbol{W}_j$ is the common column space of all $\boldsymbol{P}_i^\mathsf{T}\boldsymbol{C}_i$, and $\boldsymbol{Q}_i = \boldsymbol{P}_i\boldsymbol{W}_j$ is left orthonormal. This proves that another coupling model linking all $\boldsymbol{C}_i$ to a small latent factor $\boldsymbol{C}_c^*$ of size $R \times R$ can be obtained from the initial PARAFAC2 model with latent factor of size M. That is why, unless a specific reason is invoked to fix the size of the latent space, $\boldsymbol{C}^*$ can always be defined in $\mathbb{R}^{R\times R}$.

4.3.4 Noisy coupling

If two data sets are obtained from the same samples but with a slight change in the experimental setting, then as detailed in this section, it is possible that the shared factors $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ have some discrepancies with respect to the true underlying factor $\boldsymbol{C}^\star$ measured by a probability density. In other words,

$$\begin{array}{rcl} \boldsymbol{C}_1 &=& \boldsymbol{C}^\star + \boldsymbol{\Gamma}_1 \\ \boldsymbol{C}_2 &=& \boldsymbol{C}^\star + \boldsymbol{\Gamma}_2 \end{array}. \tag{4.77}$$

where $\boldsymbol{\Gamma}_1$ and $\boldsymbol{\Gamma}_2$ follow two independent matrix normal distributions with diagonal covariances, and columns of $\boldsymbol{C}^*$ have unit Euclidian norm. Then trying to express the coupling in a direct fashion leads to additional correlated observation noise. Indeed since the tensor product is multilinear, the following holds:

$$\begin{array}{l} \boldsymbol{\mathcal{T}}_1 = (\boldsymbol{A}_1 \otimes \boldsymbol{B}_1 \otimes \boldsymbol{C}^\star)\,\boldsymbol{\Sigma}_1 + (\boldsymbol{A}_1 \otimes \boldsymbol{B}_1 \otimes \boldsymbol{\Gamma}_1)\,\boldsymbol{\Sigma}_1 + \boldsymbol{\mathcal{E}}_1 \\ \boldsymbol{\mathcal{T}}_2 = (\boldsymbol{A}_2 \otimes \boldsymbol{B}_2 \otimes \boldsymbol{C}^\star)\,\boldsymbol{\Sigma}_2 + (\boldsymbol{A}_2 \otimes \boldsymbol{B}_2 \otimes \boldsymbol{\Gamma}_2)\,\boldsymbol{\Sigma}_2 + \boldsymbol{\mathcal{E}}_2 \end{array} \tag{4.78}$$

Now we have an exact coupled decomposition model where the noise is correlated. Moreover, this correlation depends on the factors to be estimated $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$. This can be tackled by first computing two independent decompositions, then using the estimates of uncoupled factors to write the correlation model in (4.78) to find a common representation basis under this correlated noise. This is however much trickier than the joint compression presented earlier, and an optimal solution is yet to be found.

4.4 Open problems

Understanding subject variability is a difficult topic, and it should not be surprising that some difficult issues arise from the previous results. Below we discuss possible strategies to tackle the design and estimation of the coupling model and the initialization of the coupled algorithms, although many other questions on tensor data fusion remain unanswered.

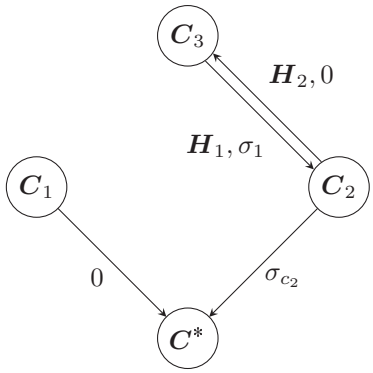


Figure 4.9: A graph representation of the coupling model (4.79).

4.4.1 Finding the coupling relationship and N tensors case: finding the graph

It is possible to generalize the Bayesian framework developed since section 4.2 to include more than two data sets and hidden factors. Dealing with multiple tensors $\boldsymbol{\mathcal{T}}_i$ coupled on the third mode means knowing at least the joint density probability $p(\boldsymbol{C}_1, \ldots, \boldsymbol{C}_N)$. Adding a hidden factor in the Bayesian framework means considering the joint probability $p\left(\boldsymbol{C}_1, \ldots, \boldsymbol{C}_N, \boldsymbol{C}^*\right)$.

However in practice things are a little more complicated. Indeed it is difficult to imagine that the coupling relationship would be available directly to the user. More likely, a coupling model is imposed by the user on the data, and the amount of fit gives insight on how well the coupling model was designed. When only two tensors are involved, there is only two possibilities for the coupling. Either it does not involve a latent factor and all what was presented above can be applied, or it does involve a latent factor $\boldsymbol{C}^*$ and the probability density that has to be known is $p\left(\boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}^*\right)$. With uninformative priors on all factors and if the factors are independent conditionally to the latent factor, this amount to determining $p\left(\boldsymbol{C}_1|\boldsymbol{C}^*\right)$ and $p\left(\boldsymbol{C}_2|\boldsymbol{C}^*\right)$.

It can be convenient to represent the coupling model as a graph, where vertices are the factors and the latent factors, and the edges represent the coupling type. The following conventions can be used:

- when only a scalar σ is provided on a directed edge, this means the two variables are coupled through Gaussian i.i.d. noise with variance σ^2.
- if a matrix $\boldsymbol{H}$ and a scalar σ are provided on a directed edge from $\boldsymbol{C}_1$ to $\boldsymbol{C}_2$, separated by a comma, then the coupling is noisy with i.i.d. noise of variance σ and $\boldsymbol{C}_1$ is transformed through $\boldsymbol{H}$.
- a double edge with another matrix $\boldsymbol{H}_2$ means the factor $\boldsymbol{C}_2$ is also transformed in the coupling relationship.
- for partially coupled factors, dotted edges are used.

This convention does not cover all the possible coupling scenarios, and more work on a pleasant graphical tool for designing coupling models is mandatory.

For instance, the graph represented in Figure 4.9 illustrates the following coupling model:

$$\begin{cases} \boldsymbol{C}_1 = \boldsymbol{C}^* \\ \boldsymbol{C}_2 = \boldsymbol{C}^* + \boldsymbol{\Gamma}_2 \\ \boldsymbol{H}_1 \boldsymbol{C}_3 = \boldsymbol{H}_2 \boldsymbol{C}_2 + \boldsymbol{\Gamma}_1 \\ \boldsymbol{\Gamma}_2 \sim \mathcal{AN}\left(\boldsymbol{0}, \sigma_{c_2}^2 \boldsymbol{I} \otimes \boldsymbol{I}\right) \\ \boldsymbol{\Gamma}_1 \sim \mathcal{AN}\left(\boldsymbol{0}, \sigma_{c_1}^2 \boldsymbol{I}\right) \\ \|\boldsymbol{C}^*\|^2 = 1 \text{ columnwise} \end{cases} . \tag{4.79}$$

Moreover, supposing Gaussian i.i.d. observation noise on the tensors, the following cost function

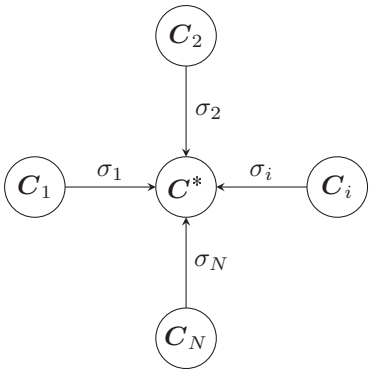


Figure 4.10: A star graph representing a coupling model where one hidden variable $\boldsymbol{C}^*$ is a centroid of all factors $\boldsymbol{C}_i$

can be derived:

$$\begin{aligned} &\tfrac{1}{\sigma_1^2}\|\boldsymbol{\mathcal{T}}_1 - \left(\boldsymbol{A}_1 \otimes \boldsymbol{B}_1 \otimes \boldsymbol{C}^*\right)\boldsymbol{I}_{R^*}\|_F^2 + \tfrac{1}{\sigma_2^2}\|\boldsymbol{\mathcal{T}}_2 - \left(\boldsymbol{A}_2 \otimes \boldsymbol{B}_2 \otimes \boldsymbol{C}_2\right)\boldsymbol{I}_{R^*}\|_F^2 \\ +\ &\tfrac{1}{\sigma_3^2}\|\boldsymbol{\mathcal{T}}_3 - \left(\boldsymbol{A}_3 \otimes \boldsymbol{B}_3 \otimes \boldsymbol{C}_3\right)\boldsymbol{I}_{R_3}\|_F^2 + \tfrac{1}{\sigma_{c_1}^2}\|\boldsymbol{H}_1\boldsymbol{C}_3 - \boldsymbol{H}_2\boldsymbol{C}_2\|_F^2 + \tfrac{1}{\sigma_{c_2}^2}\|\boldsymbol{C}_2 - \boldsymbol{C}^*\|_F^2 \end{aligned} \tag{4.80}$$

which can in turn be minimized using similar algorithms than Algorithms 16 and 17.

Another example of coupling design is given by Figure 4.10. In this scenario, a specific algorithm has been developed based on the Best Linear Unbiased Estimator (BLUE) [33], and has been used in simulations reported in section 4.2.6. Having in mind possible extensions of coupled decomposition to N data sets with large N, we may want to process the data sets in a distributed fashion. In this context, the update of the coupled factor is the only operation which cannot be parallelized through the different data sets. A straightforward option to update the coupled factors in a distributed way is to evaluate the matrices $\boldsymbol{A}_i^{\mathsf{T}}\boldsymbol{A}_i \boxdot \boldsymbol{B}_i^{\mathsf{T}}\boldsymbol{B}_i$ and $\boldsymbol{T}_{i,(3)}\left(\boldsymbol{A}_i \odot \boldsymbol{B}_i\right)$ in parallel, then apply a weighted consensus algorithm [93] with weights given by $1/\sigma_i^2$ to retrieve the sums and, finally, solve the system at each processing node. Another approximate solution is to assume that each step of uncoupled ALS generates an unbiased estimate of $\mathrm{vec}\left(\boldsymbol{C}_i\right)$ with known covariances. We can then obtain a better estimate by merging the independent estimators with a scalar BLUE. For two data sets, assuming estimation covariances $\boldsymbol{D}_i$, the scalar BLUE of $\boldsymbol{C}^\star$ is given by

$$\boldsymbol{C}^* = \frac{\sum_{i=1}^{N} \mathrm{Tr}\left(\boldsymbol{D}_i\right)^{-1}\boldsymbol{C}_i}{\sum_{i=1}^{N} \mathrm{Tr}\left(\boldsymbol{D}_i\right)^{-1}}. \tag{4.81}$$

In practice, estimation covariances are not available, and we can assume that $\mathrm{Tr}\left(\boldsymbol{D}_i\right)/\sigma_i^2 \approx \mathrm{Tr}\left(\boldsymbol{D}_j\right)/\sigma_j^2$, thus leading to a simple weighted average:

$$\boldsymbol{C}^* = \frac{\sum_{i=1}^{N} \frac{1}{\sigma_i^2}\boldsymbol{C}_i}{\sum_{i=1}^{N} \frac{1}{\sigma_i^2}}. \tag{4.82}$$

Observe from the assumption on the estimation variances that for similar noise levels on the data sets, estimation performance on all $\boldsymbol{C}_i$ must be similar. From the multilinear structure, this mainly depends on the correlation structure of the columns of the other factors. As shown in [33] and in simulations reported in section 4.2.6, this approximation gives very good estimation performance for randomly generated factors which have almost orthogonal columns.

Estimating the coupling functions Estimating the coupling model can be quite difficult, since it means finding the edges of the coupling graph, and as one can see there are quite many possibilities. Designing a coupling model for intra-subject variability is easier than for inter-subject variability, since in the first case the same motor source is expressed with small variations, but

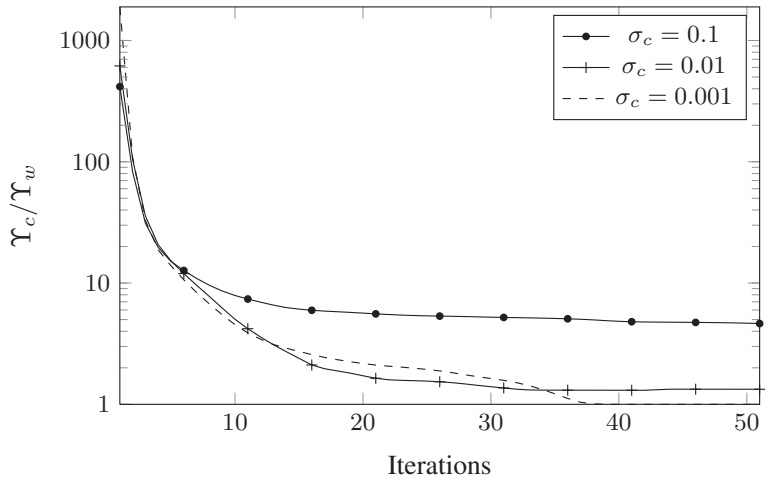


Figure 4.11: Mean of the ratio between the cost function values for the coupled CP ALS using random initialization (cold start) Υ_c and the results given by uncoupled ALS (warm start) Υ_w.

for the latter potentially very different sources and models have to account for both variations in the modelings and in the clusters of coupled data sets and parameters. That is why except exact coupled CPD, coupled models have been used mostly for studying intra-subject variability.

However once a good coupling model (or many various coupling models) has been fixed, if the coupling function are not known a priori, then they have to be estimated as well. Estimating the coupling functions is not an easy task and is heavily dependent on the admissible set $\mathcal{F}$. PARAFAC2, Shift CP, Convolutive CP and registered CP all estimate the coupling functions in an alternate fashion, but the identifiability of the coupling functions is not proven for any of these models. Moreover the tools required to compute the coupling functions in these models are very diverse, ranging from SVD to Riemmanian manifold optimization. It is not clear which kind of constraints imposed on the coupling functions leads to an identifiable coupling model. For instance, if the coupling functions are linear, say $f_i = \boldsymbol{H}_i$, then is it necessary to further parameterize $\boldsymbol{H}_i$ to be able to identify it in an alternate fashion ? Can the constraints of orthogonality imposed in PARAFAC2 be relaxed ? To give some answers to these questions is the subject of ongoing research.

4.4.2 Initialization of decomposition algorithms: Warm start vs Cold start

Here we give some simulation results to illustrate the gap in performances between coupled algorithms initialized with two independent ALS (or any other classical tensor algorithm) and two random guesses. The experiments were done for a Gaussian coupling model of the type $\boldsymbol{C}_1 = \boldsymbol{C}_2 + \boldsymbol{\Gamma}$. The ratio of the averaged cost functions are shown in Fig. 4.11 for different σ_c. The tensor dimensions are $K_1 = L_1 = M_1 = K_2 = L_2 = M_2 = 10$ and $R = 3$. The noise levels are $\sigma_1 = 0.03$ and $\sigma_2 = 0.01$ and the number of realizations is 100. Using warm start seems better than cold start. In practice, a much larger number of swamps has been observed in cold start than in warm start.

Conclusions and Perspectives

Tensor decomposition is often considered as an obscure and difficult topic, only understandable by a few well-informed researchers who try to turn other researchers into tensor-believers. Those who work with tensor decomposition methods or multiway array processing have therefore tried to extend a friendly hand to the machine learning and signal processing communities by writing more or less understandable surveys and tutorials, and providing educational speeches at various conferences. The same desire to share knowledge on this captivating and important topic has driven the content of the first two chapters which are meant to give all possible basic tools needed to understand and apply tensor decomposition techniques. The first chapter introduces the tensor product formalism that is a key element to mastering the manipulation of tensors as mathematical objects, some useful formulas and operators for tensor computations as well as exact and approximate tensor decomposition models. It stresses the absolute need for mastering the wide variety of mathematical notions surrounding tensor decomposition techniques for efficiently designing models and algorithms for mining multiway data. On the other hand, the second chapter provides a few applications of tensor decomposition techniques to environmental data mining and is meant to illustrate what is exposed in theory in the previous chapter. However, it also reveals the importance of taking into account the underlying physical processes to design efficient decomposition methods. While this might sound obvious, too often are data mining methods applied without taking the specifics of the application into account.

Mixing a good understanding of mathematical properties of tensors with a proper use of additional assumptions on the data motivated the work accomplished during the thesis. Mainly two directions have been explored: the dimensionality reduction for constrained tensors and the data fusion for tensors. Chapters 3 and 4 respectively tackle these issues while introducing a wider range of related problems and tools along the way. Again the goal is not to solely study dimensionality reduction and data fusion, but to increase the community's knowledge on possible ways to use tensor decomposition techniques to mine multiway data. That is why as often as possible, a proper theoretical framework is proposed to surround the two major contributions of the thesis that are the PROCO-ALS algorithm and the design of flexible coupling models. PROCO-ALS in an algorithm that allows the use of compressed tensors to compute the canonical polyadic decomposition, while enforcing constraints on the original unconstrained tensor. It is presently the fastest method to do so that can be adapted to tackle any constraints for which a projection on the constraint space can be computed. It is currently lacking serious convergence results. Flexible coupling models allow for two types of flexibility when modeling the coupling of tensor decompositions: a deterministic mapping of the set of parameters, and a distribution for the joint distribution of the parameters if there is a need to consider them random. This flexibility is properly motivated by a Bayesian framework, and decomposition algorithms are provided that allow for proof of concept but are not meant to be state of the art. Both PROCO-ALS and flexible coupling models are expressed respectively in the reworked frameworks of constrained decompositions and subject variability.

Interestingly, there are more questions to be asked than answers to be found in this manuscript. Among possible further research topics that may be investigated, the following perspectives will hopefully be conducted:

- to further provide models accounting for subject variability, and study in particular the estimation of the coupling model.
- to better understand the properties of PROCO-ALS with respect to the theory of contraction mappings and alternating projections.
- to develop better algorithms for the dictionary-based CPD and explore possible applications.

- to promote a healthy use of notations in the tensor community.
- to better formalize the tensorization of hyperspectral data and mine the Mars data from the Mars Reconnaissance Orbiter mission.

Finally, if this manuscript serves the purpose of clarifying any aspect of multiway data mining to at least one single non-expert reader, then this alone would be a great satisfaction. For expert readers, most of the content of the above pages can be found in following publications that occurred during the PhD [9, 21, 22, 29, 30, 32–34, 99, 116–119].

Glossary

Objects and Variables

λ	scalar
$\boldsymbol{v}$	vector
$\boldsymbol{M}$	matrix
$\boldsymbol{\mathcal{T}}$	tensor or multiway array
v_i	coefficient i of vector $\boldsymbol{v}$
M_{ij}	coefficient at row i and at column j of matrix $\boldsymbol{M}$
T_{klm}	coefficient klm of multiway array $\boldsymbol{\mathcal{T}}$
$\boldsymbol{M}_i$	matrix of index i among N matrices
$\boldsymbol{\mathcal{T}}_i$	tensor of index i among N tensors
$\boldsymbol{m}_i$	column i of matrix $\boldsymbol{M}$; vector of index i if $\boldsymbol{M}$ not specified
$\boldsymbol{m}_i^{(j)}$	column i of matrix $\boldsymbol{M}_j$
$\boldsymbol{T}_{i,(j)}$	jth unfolding of tensor $\boldsymbol{\mathcal{T}}_i$
$\boldsymbol{\Sigma}_R$	diagonal tensor with R non-zero values
$\boldsymbol{I}_R$	diagonal tensor of ones with R non-zero values
$\delta_{i,j}$	Kronecker symbol, equals to 1 iff i=j, else equals 0

Operators

$\otimes$	tensor product
$\boxtimes$	Kronecker product
$\odot$	Khatri-Rao product
$\boxdot$	Hadamard product
$\circ$	outer product of vector; composition of functions
$\bullet_i$	contraction on the ith mode
$N!$	factorial N
$\mathrm{Tr}\left(\boldsymbol{\mathcal{T}}\right)$	trace of tensor $\boldsymbol{\mathcal{T}}$
${}^{\dagger}\boldsymbol{M}$	left inverse of matrix $\boldsymbol{M}$
vec	vectorization as in section 1.1.1
$[\boldsymbol{a}, \boldsymbol{b}]$	horizontal concatenation
$[\boldsymbol{a}; \boldsymbol{b}]$	vertical concatenation
$\boldsymbol{M}^{\mathsf{T}}$	transpose of matrix $\boldsymbol{M}$
$[x]^+$	positive part of x
$\lvert\boldsymbol{\mathcal{T}}\rvert$	determinant of tensor $\boldsymbol{\mathcal{T}}$
$\lfloor x \rfloor$	integer part of real x

Probabilistic tools

$X \sim \mathcal{D}$	random variable X follows distribution $\mathcal{D}$
$\mathcal{N}\left(m, \sigma^2\right)$	Gaussian/normal distribution of mean m and variance σ^2
$\mathcal{AN}\left(\boldsymbol{\mathcal{T}}, \boldsymbol{\Gamma}\right)$	array normal distribution of mean $\boldsymbol{\mathcal{T}}$ and covariance $\boldsymbol{\Sigma}$
X^*	latent variable related to random variable X
$X\lvert Y$	random variable X conditioned by random variable Y (or deterministic Y abusively).
$p_\theta(X)$	probability density of random variable X depending on θ

Bibliography

[1] E. Acar, R. Bro, and A.K. Smilde. Data fusion in metabolomics using coupled matrix and tensor factorizations. *Proceedings of the IEEE*, 103(9):1602–1620, 2015.

[2] E. Acar, D.M. Dunlavy, T.G. Kolda, and M. Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, 2011.

[3] E. Acar, T.G. Kolda, and D.M. Dunlavy. All-at-once optimization for coupled matrix and tensor factorizations. 2011.

[4] E. Acar, A.J. Lawaetz, M.A. Rasmussen, and R. Bro. Structure-revealing data fusion model with applications in metabolomics. In *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, pages 6023–6026. IEEE, 2013.

[5] Evrim Acar and Bülent Yener. Unsupervised multiway data analysis: A literature survey. *IEEE transactions on knowledge and data engineering*, 21(1):6–20, 2009.

[6] R. Ballester-Ripoll, S.K. Suter, and R. Pajarola. Analysis of tensor approximation for compression-domain volume visualization. *Computers & Graphics*, 47:34–47, 2015.

[7] H. Becker. *Débruitage, séparation et localisation de sources EEG dans le contexte de l'épilepsie*. PhD thesis, 2014. Thèse de doctorat dirigée par Comon, P. et Albera, L. Automatique, traitement du signal et des images Nice 2014.

[8] J.M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot. Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, 5(2):354–379, 2012.

[9] M. Boizard, R. Boyer, G. Favier, J.E. Cohen, and P. Comon. Performance estimation for tensor cp decomposition with structured factors. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 3482–3486. IEEE, 2015.

[10] C.C. Borel and S.A.W. Gerstl. Nonlinear spectral mixing models for vegetative and soil surfaces. *Remote sensing of environment*, 47(3):403–416, 1994.

[11] Nicolas Bourbaki. *Algebre linéaire*. Hermann, 1947.

[12] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[13] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[14] J.P. Boyle and R.L. Dykstra. A method for finding projections onto the intersection of convex sets in hilbert spaces. In *Advances in order restricted statistical inference*, pages 28–47. Springer, 1986.

[15] R. Bro. Parafac. tutorial and applications. *Chemometrics and intelligent laboratory systems*, 38(2):149–171, 1997.

[16] R. Bro. *Multi-way analysis in the food industry: models, algorithms, and applications*. PhD thesis, University of Amsterdam, The Netherlands, 1998.

[17] R. Bro, C.A. Andersson, and H.A.L. Kiers. Parafac2-part ii. modeling chromatographic data with retention time shifts. *Journal of Chemometrics*, 13(3-4):295–309, 1999.

[18] R. Bro and S. De Jong. A fast non-negativity-constrained least squares algorithm. *Journal of chemometrics*, 11(5):393–401, 1997.

[19] R. Bro, R.A. Harshman, N.D. Sidiropoulos, and M.E. Lundy. Modeling multi-way data with linearly dependent loadings. *Journal of Chemometrics*, 23(7-8):324–340, 2009.

[20] R. Bro and H.A.L. Kiers. A new efficient method for determining the number of components in parafac models. *Journal of chemometrics*, 17(5):274–286, 2003.

[21] R. Cabral-Farias, J.E. Cohen, and P. Comon. Exploring multimodal data fusion through joint decompositions with flexible couplings. *Signal Processing, IEEE Transactions on*, to appear, 2016.

[22] R. Cabral Farias, J.E. Cohen, C. Jutten, and P. Comon. Joint decompositions with flexible couplings. In *Latent Variable Analysis and Signal Separation*, pages 119–126. Springer, 2015.

[23] C.F. Caiafa and A. Cichocki. Multidimensional compressed sensing and their applications. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 3(6):355–380, 2013.

[24] F. Caland, S. Miron, D. Brie, and C. Mustin. A blind sparse approach for estimating constraint matrices in paralind data models. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 839–843. IEEE, 2012.

[25] J.D. Carroll and J-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[26] J.D. Carroll, S. Pruzansky, and J.B. Kruskal. Candelinc: A general approach to multidimensional analysis of many-way arrays with linear constraints on parameters. *Psychometrika*, 45(1):3–24.

[27] L. Chiantini and G. Ottaviani. On generic identifiability of 3-tensors of small rank. *SIAM Journal on Matrix Analysis and Applications*, 33(3):1018–1037, 2012.

[28] A. Cichocki, R. Zdunek, A.H. Phan, and S. Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[29] J. Cohen and P. Comon. On almost sure identifiability of non multilinear tensor decomposition. In *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*, pages 2245–2249. IEEE, 2014.

[30] J.E. Cohen. About notations in multiway array processing. *arXiv preprint arXiv:1511.01306*, 2015.

[31] J.E. Cohen, R. Cabral Farias, and P. Comon. Analyse de grandes donnees ten- sorielles couplees. In *XXVeme colloque GRETSI 2015*, Lyon, France.

[32] J.E. Cohen, R. Cabral Farias, and P. Comon. Fast decomposition of large nonnegative tensors. *Signal Processing Letters, IEEE*, 22(7):862–866, 2015.

[33] J.E. Cohen, R. Cabral Farias, and P. Comon. Joint tensor compression for coupled canonical polyadic decompositions. In *2016 24th European Signal Processing Conference (EUSIPCO 2016)*, Budapest, Hungary, August 2016.

[34] J.E. Cohen, P. Comon, and X. Luciani. Correcting inner filter effects, a non multilinear tensor decomposition method. *Chemometrics and Intelligent Laboratory Systems*, 150:29–40, 2016.

[35] J.E. Cohen, K. Usevich, and P. Comon. A Tour of Constrained Tensor Canonical Polyadic Decomposition. working paper or preprint, April 2016.

[36] P.L. Combettes and J-C. Pesquet. Primal-dual splitting algorithm for solving inclusions with mixtures of composite, lipschitzian, and parallel-sum type monotone operators. *Set-Valued and variational analysis*, 20(2):307–330, 2012.

[37] P. Comon. Tensors: a brief introduction. *IEEE Signal Processing Magazine*, 31(3):44–53, 2014.

[38] P. Comon and C. Jutten. *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press, 2010.

[39] L. Condat. A generic proximal algorithm for convex optimization—application to total variation minimization. *Signal Processing Letters, IEEE*, 21(8):985–989, 2014.

[40] M. Congedo. *EEG Source Analysis*. Accreditation to supervise research, Université de Grenoble, October 2013.

[41] M. Congedo, M. Goyat, N. Tarrin, G. Ionescu, L. Varnet, B. Rivet, R. Phlypo, N. Jrad, M. Acquadro, and C. Jutten. "Brain Invaders": a prototype of an open-source p300-based video game working with the openvibe platform. In *5th International Brain-Computer Interface Conference 2011 (BCI 2011)*, pages 280–283, 2011.

[42] L. De Lathauwer. Decompositions of a higher-order tensor in block terms-part ii: definitions and uniqueness. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1033–1066, 2008.

[43] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.

[44] L. De Lathauwer and D. Nion. Decompositions of a higher-order tensor in block terms-part iii: Alternating least squares algorithms. *SIAM journal on Matrix Analysis and Applications*, 30(3):1067–1083, 2008.

[45] L. De Lathauwer and J. Vandewalle. Dimensionality reduction in higher-order signal processing and rank-$(R_1, R_2, \ldots, R_n)$ reduction in multilinear algebra. *Linear Algebra and its Applications*, 391:31–55, 2004.

[46] Lieven De Lathauwer. Blind separation of exponential polynomials and the decomposition of a tensor in rank-(l_r,l_r,1) terms. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1451–1474, 2011.

[47] V. De Silva and L.H. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1084–1127, 2008.

[48] C-A. Deledalle, L. Denis, G. Poggi, F. Tupin, and L. Verdoliva. Exploiting patch similarity for sar image processing: the nonlocal paradigm. *Signal Processing Magazine, IEEE*, 31(4):69–78, 2014.

[49] I. Domanov and L. De Lathauwer. On the uniqueness of the canonical polyadic decomposition of third-order tensors—part ii: Uniqueness of the overall decomposition. *SIAM Journal on Matrix Analysis and Applications*, 34(3):876–903, 2013.

[50] I. Domanov and L. De Lathauwer. Canonical polyadic decomposition of third-order tensors: reduction to generalized eigenvalue decomposition. *SIAM Journal on Matrix Analysis and Applications*, 35(2):636–660, 2014.

[51] Ignat Domanov and Lieven De Lathauwer. Generic uniqueness of a structured matrix factorization and applications in blind source separation. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):701–711, 2016.

[52] L. Drumetz, S. Henrot, M.A. Veganzones, J. Chanussot, and C. Jutten. Blind hyperspectral unmixing using an extended linear mixing model to address spectral variability. In *IEEE Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS 2015)*, 2015.

[53] J.W. Emsley, J. Feeney, and L.H. Sutcliffe. *High resolution nuclear magnetic resonance spectroscopy*. Pergamon Press, 1965.

[54] B. Ermiş, E. Acar, and A. T. Cemgil. Link prediction via generalized coupled tensor factorisation. *arXiv preprint arXiv:1208.6231*, 2012.

[55] N. Gillis et al. *Nonnegative matrix factorization: Complexity, algorithms and applications*. PhD thesis, UCL, 2011.

[56] J.D. Gorman and A.O. Hero. Lower bounds for parametric estimation with constraints. *Information Theory, IEEE Transactions on*, 36(6):1285–1301, 1990.

[57] W. Hackbusch. *Tensor spaces and numerical tensor calculus*, volume 42. Springer Science & Business Media, 2012.

[58] B. Hapke. *Theory of reflectance and emittance spectroscopy*. Cambridge University Press, 2012.

[59] R.A. Harshman. Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-modal factor analysis. 1970.

[60] R.A. Harshman. Parafac2: Mathematical and technical notes. *UCLA working papers in phonetics*, 22(3044):122215, 1972.

[61] R.A. Harshman, S. Hong, and M.E. Lundy. Shifted factor analysis—Part I: Models and properties. *Journal of chemometrics*, 17(7):363–378, 2003.

[62] R.A. Harshman and M.E. Lundy. Data preprocessing and the extended PARAFAC model. *Research Methods for Multimode Data Analysis*, pages 216–284, 1984.

[63] C. Hildreth. A quadratic programming procedure. *Naval research logistics quarterly*, 4(1):79–85, 1957.

[64] F.L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1):164–189, 1927.

[65] G. Hollander, P. Dreesen, M. Ishteva, and J. Schoukens. Weighted tensor decomposition for approximate decoupling of multivariate polynomials. *arxiv e-prints*, January 2016.

[66] S. Hong and R.A. Harshman. Shifted factor analysis—Part II: algorithms. *Journal of chemometrics*, 17(7):379–388, 2003.

[67] S. Hong and R.A. Harshman. Shifted factor analysis—part iii: N-way generalization and application. *Journal of chemometrics*, 17(7):389–399, 2003.

[68] K. Huang, N.D. Sidiropoulos, and A.P. Liavas. A flexible and efficient algorithmic framework for constrained matrix and tensor factorization. *arXiv preprint arXiv:1506.04209*, 2015.

[69] B. Jørgensen. *The theory of exponential dispersion models and analysis of deviance*. Number 51. CNPQ, IMPA (Brazil), 1992.

[70] H.A.L. Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of chemometrics*, 14(3):105–122, 2000.

[71] H.A.L. Kiers, J.M.F. Ten Berge, and R. Bro. Parafac2-part i. a direct fitting algorithm for the parafac2 model. *Journal of Chemometrics*, 13(3-4):275–294, 1999.

[72] T.G. Kolda and B.W. Bader. Tensor decompositions and applications. *SIAM rev.*, 51(3):455–500, 2009.

[73] L. Korczowski, M. Congedo, and C. Jutten. Single-trial classification of multi-user p300-based brain-computer interface using riemannian geometry. In *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, pages 1769–1772. IEEE, 2015.

[74] S. Kotz and N.L. Johnson. *Breakthroughs in statistics: Foundations and basic theory*. Springer Science & Business Media, 2012.

[75] Wim P Krijnen, Theo K Dijkstra, and Alwin Stegeman. On the non-existence of optimal solutions and the occurrence of "degeneracy" in the candecomp/parafac model. *Psychometrika*, 73(3):431–439, 2008.

[76] D.J. Krusienski, E.W. Sellers, F. Cabestaing, S. Bayoudh, D.J. McFarland, T.M. Vaughan, and J.R. Wolpaw. A comparison of classification techniques for the p300 speller. *Journal of neural engineering*, 3(4):299, 2006.

[77] J.B. Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear algebra and its applications*, 18(2):95–138, 1977.

[78] C. L. Lawson and R. J. Hanson. *Solving least squares problems*, volume 15. SIAM, 1995.

[79] D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[80] A.P. Liavas and N.D. Sidiropoulos. Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers. *Signal Processing, IEEE Transactions on*, 63(20):5450–5463, 2015.

[81] L.H. Lim and P. Comon. Nonnegative approximations of nonnegative tensors. *Journal of chemometrics*, 23(7-8):432–441, 2009.

[82] L.H. Lim and P. Comon. Multiarray signal processing: Tensor decomposition meets compressed sensing. *Comptes Rendus Mecanique*, 338(6):311–320, 2010.

[83] L.H. Lim and P. Comon. Blind multilinear identification. *Information Theory, IEEE Transactions on*, 60(2):1260–1280, 2014.

[84] X. Liu and N.D. Sidiropoulos. Cramér-Rao lower bounds for low-rank decomposition of multidimensional arrays. *IEEE Trans. Signal Process.*, 49(9):2074–2086, 2001.

[85] X. Luciani, S. Mounier, R. Redon, and A. Bois. A simple correction method of inner filter effects affecting feem and its application to the parafac decomposition. *Chemometrics and Intelligent Laboratory Systems*, 96(2):227–238, 2009.

[86] K. Madsen, H.B. Nielsen, and O. Tingleff. Methods for non-linear least squares problems. 2004.

[87] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F.R. Bach. Supervised dictionary learning. In *Advances in neural information processing systems*, pages 1033–1040, 2009.

[88] S.G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on*, 41(12):3397–3415, 1993.

[89] M. Mørup, L.K. Hansen, S.M. Arnfred, L.H. Lim, and K.H. Madsen. Shift-invariant multilinear decomposition of neuroimaging data. *NeuroImage*, 42(4):1439–1450, 2008.

[90] M. Mørup, L.K. Hansen, and K.H. Madsen. Modeling latency and shape changes in trial based neuroimaging data. In *Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on*, pages 439–443. IEEE, 2011.

[91] Morten Mørup, Lars Kai Hansen, and Sidse M Arnfred. Algorithms for sparse nonnegative tucker decompositions. *Neural computation*, 20(8):2112–2131, 2008.

[92] D. Muti and S. Bourennane. Survey on tensor signal algebraic filtering. *Signal Processing*, 87(2):237–249, 2007.

[93] R. Olfati-Saber, A. Fax, and R. M. Murray. Consensus and cooperation in networked multiagent systems. *Proc. IEEE*, 95(1):215–233, January 2007.

[94] R.D. Pascual-Marqui et al. Standardized low-resolution brain electromagnetic tomography (sloreta): technical details. *Methods Find Exp Clin Pharmacol*, 24(Suppl D):5–12, 2002.

[95] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[96] Nelly Pustelnik, Jean-Christophe Pesquet, and Caroline Chaux. Relaxing tight frame condition in parallel proximal methods for signal restoration. *Signal Processing, IEEE Transactions on*, 60(2):968–973, 2012.

[97] C.R. Rao. Information and accuracy attainable in the estimation of statistical parameters. *Bull Calcutta. Math. Soc.*, 37:81–91, 1945.

[98] Å. Rinnan and C.M. Andersson. Handling of first-order rayleigh scatter in parafac modelling of fluorescence excitation–emission data. *Chemometrics and intelligent laboratory systems*, 76(1):91–99, 2005.

[99] B. Rivet and J.E. Cohen. Modeling time warping in tensor decomposition. In *2016 IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM 2016)*, Rio de Janeiro, Brazil, July 2016.

[100] S. Sahnoun, E-H. Djermoune, D. Brie, and P. Comon. A simultaneous sparse approximation method for multidimensional harmonic retrieval. *arXiv preprint arXiv:1507.02075*, 2015.

[101] L. Schwartz, Y. Bamberger, and J-P. Bourguignon. Les tenseurs. 1977.

[102] N. Seichepine, S. Essid, C. Fevotte, and O. Cappé. Soft nonnegative matrix co-factorization. *IEEE Trans. Sig. Process.*, 62(22):5940–5949, Nov 2014.

[103] A.P. Singh and G.J. Gordon. Relational learning via collective matrix factorization. In *Proc. 14th ACM SIGKDD Conf.*, pages 650–658. ACM, 2008.

[104] M. Sørensen and L. De Lathauwer. Coupled canonical polyadic decompositions and (coupled) decompositions in multilinear rank-$(L_{r,n}, L_{r,n}, 1)$ terms. Part I: Uniqueness. Technical report, KU Leuven (Belgium), 2013.

[105] M. Sørensen, I. Domanov, D. Nion, and L. De Lathauwer. Coupled canonical polyadic decompositions and (coupled) decompositions in multilinear rank-$(L_{r,n}, L_{r,n}, 1)$ terms. Part II: Algorithms. Technical report, KU Leuven (Belgium), 2013.

[106] Mikael Sørensen and Lieven De Lathauwer. Blind signal separation via tensor decomposition with vandermonde factor: Canonical polyadic decomposition. *IEEE Transactions on Signal Processing*, 61(22):5507–5519, 2013.

[107] Mikael Sørensen, Lieven De Lathauwer, Pierre Comon, Sylvie Icart, and Luc Deneire. Canonical polyadic decomposition with a columnwise orthonormal factor matrix. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1190–1213, 2012.

[108] A. Stegeman and A.L.F. De Almeida. Uniqueness conditions for constrained three-way factor decompositions with linearly dependent loadings. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1469–1490, 2009.

[109] A. Stegeman and N.D. Sidiropoulos. On Kruskal's uniqueness condition for the candecomp/parafac decomposition. *Linear Algebra and its applications*, 420(2):540–552, 2007.

[110] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[111] P. Tichavsky, A. H. Phan, and Z. Koldovsky. Cramér-Rao-induced bounds for candecomp/parafac tensor decomposition. *Signal Processing, IEEE Transactions on*, 61(8):1986–1997, 2013.

[112] M.E. Timmerman and H.A.L. Kiers. Three-way component analysis with smoothness constraints. *Computational statistics & data analysis*, 40(3):447–470, 2002.

[113] L.R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[114] H.L. Van Trees. *Detection, estimation, and modulation theory*. John Wiley & Sons, 2004.

[115] H.L. Van Trees and K.L. Bell. *Bayesian Bounds for Parameter Estimation and Nonlinear Filtering/Tracking*. Wiley-IEEE Press, 2007.

[116] M.A. Veganzones, J.E. Cohen, R. Cabral Farias, J. Chanussot, and P. Comon. Nonnegative tensor cp decomposition of hyperspectral data. *Geoscience and Remote Sensing, IEEE Transactions on*, PP(99):1–12, 2015.

[117] M.A. Veganzones, J.E. Cohen, R. Cabral Farias, R. Marrero, J. Chanussot, and P. Comon. Multilinear spectral unmixing of hyperspectral multiangle images. In *Signal Processing Conference (EUSIPCO), 2015 23rd European*, pages 744–748. IEEE, 2015.

[118] M.A. Veganzones, J.E. Cohen, R. Cabral Farias, K. Usevich, L. Drumetz, J. Chanussot, and P. Comon. Canonical polyadic decomposition of hyperspectral patch tensors. In *2016 24th European Signal Processing Conference (EUSIPCO 2016)*, Budapest, Hungary, August 2016.

[119] M.A. Veganzones, S. Douté, J.E. Cohen, R. Cabral Farias, J. Chanussot, and P. Comon. Nonnegative cp decomposition of multiangle hyperspectral data: a case study on CRISM observations of martian icy surface. In *IEEE Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS 2016)*, 2016.

[120] N Vervliet, O Debals, L Sorber, M Van Barel, and L De Lathauwer. Tensorlab v3. 0. *available online, URL: www. tensorlab. net*, 2016.

[121] W. Wang and M.A. Carreira-Perpinán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv preprint arXiv:1309.1541*, 2013.

[122] H Whitney et al. Tensor products of abelian groups. *Duke Mathematical Journal*, 4(3):495–528, 1938.

[123] B. Yang, X. Fu, and N.D. Sidiropoulos. Learning from hidden traits: Joint factor analysis and latent clustering. *arXiv preprint arXiv:1605.06711*, 2016.

[124] K.Y. Yılmaz, A.T. Cemgil, and U. Simsekli. Generalised coupled tensor factorisation. In *Adv. Neural. Inf. Process. Syst.*, pages 2151–2159, 2011.

[125] Y.K. Yilmaz and A.T. Cemgil. Alpha/beta divergences and tweedie models. *arXiv preprint arXiv:1209.4280*, 2012.

[126] Yu Zhang, Guoxu Zhou, Qibin Zhao, Andrzej Cichocki, and Xingyu Wang. Fast nonnegative tensor factorization based on accelerated proximal gradient and low-rank approximation. *Neurocomputing*, 198:148–154, 2016.

Appendix A

Trace and Frobenius norm of tensors

A common mistake about the trace is thinking that it always amounts to the sum of the diagonal values of arrays. For instance, given a three-way array $\boldsymbol{\mathcal{T}}$ in $\mathbb{R}^{K\times K\times K}$, a naive definition of the trace would be $\mathrm{Tr}\left(\boldsymbol{\mathcal{T}}\right) = \sum T_{iii}$. However this would be a complete misunderstanding of what the trace operator is. Indeed, the trace operator acts on linear mappings of a vector space, not on the vector space itself, but $\boldsymbol{\mathcal{T}}$ is a vector of the array vector space. Here is an intrinsic definition of the trace operator:

Definition 18 *Let $\mathscr{E}$ a vector space, let $\mathscr{E}^*$ the space of linear forms on $\mathscr{E}$. The bilinear mapping $\boldsymbol{e}\otimes f = f(\bullet)\boldsymbol{e}$ on $\mathscr{E}\otimes\mathscr{E}^*$ defines a tensor product space $(\mathscr{E}\otimes\mathscr{E}^*,\otimes)$. Then for all $(\boldsymbol{e},f)$ in $\mathscr{E}\times\mathscr{E}^*$,*

$$\mathrm{Tr}\left(\boldsymbol{e}\otimes f\right) = f(\boldsymbol{e}). \tag{A.1}$$

For simplicity purpose, during hand written computation, and since the trace is a scalar product, it is possible to write $\mathrm{Tr}\left(\boldsymbol{e}\otimes f\right) = \langle \boldsymbol{e}|\boldsymbol{f}\rangle$. Since any linear operator $\boldsymbol{U}\in\mathbb{R}^{N\times N}$ can be written as a sum of orthogonal R rank one linear mappings $\boldsymbol{a}_r\otimes\boldsymbol{b}_r$, the trace for matrices is indeed the sum of diagonal coefficients of $\boldsymbol{U}$:

$$\mathrm{Tr}\left(\boldsymbol{U}\right) = \sum_{r=1}^{R}\boldsymbol{b}_r^{\mathsf{T}}\boldsymbol{a}_r = \sum_{i=1}^{N}\sum_{r=1}^{R}[\boldsymbol{a}_r]_i[\boldsymbol{b}_r]_i = \sum_{i=1}^{N}U_{ii}. \tag{A.2}$$

In finite dimensions and for $\mathscr{E} = \mathbb{R}^K\otimes\mathbb{R}^L\otimes\mathbb{R}^M$, supposing the trace is linear, this definition can be rewritten in a slightly more familiar manner:

$$\begin{array}{rccc} & \mathcal{L}\left(\mathbb{R}^K,\mathbb{R}^K\right)\otimes\mathcal{L}\left(\mathbb{R}^L,\mathbb{R}^L\right)\otimes\mathcal{L}\left(\mathbb{R}^M,\mathbb{R}^M\right) & \rightarrow & \mathbb{R} \\ \mathrm{Tr}: & (\boldsymbol{a}\otimes\boldsymbol{b}\otimes\boldsymbol{c})\,(\boldsymbol{d}\otimes\boldsymbol{e}\otimes\boldsymbol{f})^{\mathsf{T}} & \mapsto & (\boldsymbol{d}^{\mathsf{T}}\boldsymbol{a})(\boldsymbol{e}^{\mathsf{T}}\boldsymbol{b})(\boldsymbol{f}^{\mathsf{T}}\boldsymbol{c}) \end{array} \tag{A.3}$$

and a formula similar to matrix trace can be obtained for square operators acting on arrays:

$$\mathrm{Tr}\left(\boldsymbol{U}\otimes\boldsymbol{V}\otimes\boldsymbol{W}\right) = \left(\sum_{k=1}^{K}\boldsymbol{U}_{kk}\right)\left(\sum_{l=1}^{L}\boldsymbol{V}_{ll}\right)\left(\sum_{m=1}^{M}\boldsymbol{W}_{mm}\right). \tag{A.4}$$

This definition implies that $\mathrm{Tr}\left(\boldsymbol{\mathcal{T}}\right)$ is not well-defined since $\boldsymbol{\mathcal{T}}$ is a vector, and the trace operator can only be applied to operators. For the matrix case, $\mathrm{Tr}\left(\boldsymbol{U}\right)$ is the trace of the linear operator $\boldsymbol{U}$ acting on vectors of some vector space.

The trace operator is useful in a wide range of situations since it is a scalar product. In quantum mechanics, the trace stands for a physical measurement of a quantum quantity. In multilinear algebra, it can be used to compute the norm of a tensor:

Definition 19 *Let $\boldsymbol{\mathcal{T}} = \sum_{r=1}^{R}\boldsymbol{a}_r\otimes\boldsymbol{b}_r\otimes\boldsymbol{c}_r$ be a tensor in $\mathbb{R}^K\otimes\mathbb{R}^L\otimes\mathbb{R}^M$. Denote by $\boldsymbol{\mathcal{T}}^* = \sum_{r=1}^{R}\boldsymbol{a}_r^{\mathsf{T}}\otimes\boldsymbol{b}_r^{\mathsf{T}}\otimes\boldsymbol{c}_r^{\mathsf{T}}$ the adjoint of $\boldsymbol{\mathcal{T}}$ in the dual space $\mathcal{L}\left(\mathbb{R}^K\otimes\mathbb{R}^L\otimes\mathbb{R}^M,\mathbb{R}\right)$. Then the Frobenius norm of $\boldsymbol{\mathcal{T}}$ is defined as follows:*

$$\|\boldsymbol{\mathcal{T}}\|_F^2 = \mathrm{Tr}\left(\boldsymbol{\mathcal{T}}\otimes\boldsymbol{\mathcal{T}}^*\right) = \langle\boldsymbol{\mathcal{T}}|\boldsymbol{\mathcal{T}}\rangle = \sum_{r,p}^{R}(\boldsymbol{a}_r^{\mathsf{T}}\boldsymbol{a}_p)(\boldsymbol{b}_r^{\mathsf{T}}\boldsymbol{b}_p)(\boldsymbol{c}_r^{\mathsf{T}}\boldsymbol{c}_p) = \sum_{ijk}T_{ijk}^2. \tag{A.5}$$

Here the definition of the Frobenius norm is detailed for real finite dimension tensors, but a generalization to any kind of vector space is trivial.

Another interesting concept is the partial trace defined in section 1.3.2. Partial trace allows to study a tensor mode-wise without resorting to matricization. There is a simple relationship between the partial trace and the trace:

$$\mathrm{Tr}\left(\boldsymbol{U}\otimes\boldsymbol{V}\right)=\mathrm{Tr}_1\left(\boldsymbol{U}\right)\mathrm{Tr}_2\left(\boldsymbol{V}\right). \tag{A.6}$$

While matricization is intuitive, using partial traces for theoretical computations yields wider impact. For instance, it is possible to compute the gradient of the norm of a linearly modified tensor using only partial traces:

Proof of proposition 3 Let us prove that

$$\frac{\partial\gamma(\boldsymbol{U})}{\partial\boldsymbol{U}}=\frac{\partial\|\left(\boldsymbol{U}\otimes\boldsymbol{V}\otimes\boldsymbol{W}\right)\boldsymbol{\mathcal{T}}\|}{\partial\boldsymbol{U}}=\boldsymbol{U}\mathrm{Tr}_{2,3}\left(\left(\boldsymbol{I}\otimes\boldsymbol{V}\otimes\boldsymbol{W}\right)\left(\boldsymbol{\mathcal{T}}\left(\boldsymbol{I}\otimes\boldsymbol{V}\otimes\boldsymbol{W}\right)\boldsymbol{\mathcal{T}}\right)^{*}\right). \tag{A.7}$$

The differential $D_\gamma(\boldsymbol{H},\boldsymbol{U})$ is of the form $\mathrm{Tr}_1\left(\nabla_\gamma\boldsymbol{H}^*\right)$ since the derivative is taken with respect to the subset of variables $\boldsymbol{U}$. So to obtain the gradient of the Frobenius norm γ, it is sufficient to compute the differential of γ:

$$\begin{aligned}\|\left(\left(\boldsymbol{U}+\boldsymbol{H}\otimes\boldsymbol{V}\otimes\boldsymbol{W}\right)\boldsymbol{\mathcal{T}}\right\| &= \langle\left(\left(\boldsymbol{U}+\boldsymbol{H}\right)\otimes\boldsymbol{V}\otimes\boldsymbol{W}\right)\boldsymbol{\mathcal{T}}\mid\left(\left(\boldsymbol{U}+\boldsymbol{H}\right)\otimes\boldsymbol{V}\otimes\boldsymbol{W}\right)\boldsymbol{\mathcal{T}}\rangle\\ &= \|\left(\boldsymbol{U}\otimes\boldsymbol{V}\otimes\boldsymbol{W}\right)\boldsymbol{\mathcal{T}}\|_F^2+2\,\mathrm{Tr}\left(\left(\boldsymbol{U}\otimes\boldsymbol{V}\otimes\boldsymbol{W}\right)\boldsymbol{\mathcal{T}}\boldsymbol{\mathcal{T}}^*\left(\boldsymbol{H}\otimes\boldsymbol{V}\otimes\boldsymbol{W}\right)^*\right)\\ &\quad+o(\boldsymbol{H})\end{aligned}. \tag{A.8}$$

Further using the fact that $\mathrm{Tr}\left(\boldsymbol{U}\otimes\boldsymbol{V}\right)=\mathrm{Tr}_1\left(\mathrm{Tr}_2\left(\boldsymbol{U}\otimes\boldsymbol{V}\right)\right)$ and that $\mathrm{Tr}_2\left(\left(\boldsymbol{U}\otimes\boldsymbol{I}\right)\boldsymbol{\mathcal{T}}\right)=\boldsymbol{U}\mathrm{Tr}_2\left(\boldsymbol{\mathcal{T}}\right)$, the differential of γ can be expressed as follows:

$$D_\gamma(\boldsymbol{H},\boldsymbol{U})=\mathrm{Tr}_1\left[\boldsymbol{U}\mathrm{Tr}_2\left[\left(\boldsymbol{I}\otimes\boldsymbol{V}\otimes\boldsymbol{W}\right)\boldsymbol{\mathcal{T}}\boldsymbol{\mathcal{T}}^*\left(\boldsymbol{I}\otimes\boldsymbol{V}\otimes\boldsymbol{W}\right)^*\right]\boldsymbol{H}^*\right]. \tag{A.9}$$

This means that the gradient of γ with respect to $\boldsymbol{U}$ is simply

$$\boldsymbol{U}\mathrm{Tr}_2\left[\left(\boldsymbol{I}\otimes\boldsymbol{V}\otimes\boldsymbol{W}\right)\boldsymbol{\mathcal{T}}\boldsymbol{\mathcal{T}}^*\left(\boldsymbol{I}\otimes\boldsymbol{V}\otimes\boldsymbol{W}\right)^*\right] \tag{A.10}$$

which coincides with the gradient obtained when the tensor in an unfolded array. ▫

Appendix B

Cramér Rao bound for approximate CPD

The Cramér-Rao Bound (CRB) is a lower bound on the variance of unbiased estimators of parameters in probabilistic models. It is computed using the Fisher Information Matrix (FIM) defined as follows:

$$\boldsymbol{F} = \mathbb{E}\left[\frac{\partial \log(p(\boldsymbol{x}|\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}} \frac{\partial \log(p(\boldsymbol{x}|\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}}^T\right] \tag{B.1}$$

where $p(\boldsymbol{x}|\boldsymbol{\theta})$ is the probability density of some data $\boldsymbol{x}$ depending on a deterministic parameter vector $\boldsymbol{\theta}$. Under regularity conditions [114], the inverse of the FIM lower bounds the covariance of any unbiased estimator $\hat{\boldsymbol{\theta}}$ of the vector of parameter $\boldsymbol{\theta}$:

$$\left[\boldsymbol{F}^{-1}\right]_{ij} \leqslant \mathbb{E}\left[\hat{\theta}_i \hat{\theta}_j\right] \tag{B.2}$$

In the context of approximate CPD with Gaussian noise, a probabilistic model on the data can be written as follows:

$$\begin{cases} \boldsymbol{\mathcal{T}} = (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\boldsymbol{\mathcal{I}}_R + \boldsymbol{\mathcal{E}} \\ \text{vec}(\boldsymbol{\mathcal{E}}) \sim \mathcal{N}\left(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_{KLM}\right) \\ A_{1i} = B_{1i} = 1 \ \forall i \in [1, R] \end{cases} \tag{B.3}$$

The constraints imposed on the first row of $\boldsymbol{A}$ and $\boldsymbol{B}$ remove the scaling ambiguity, thus allowing the Fisher information matrix to be invertible.

Now let us derive the CRB for the approximate CPD. Computation of the CRB in this context goes back to [**?**], here another way of computing the CRB is suggested.

First, note that the CRB is written for a vector of parameters, so that the derivative of the log-likelihood $\frac{1}{\sigma^2}\|\boldsymbol{\mathcal{T}} - (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\boldsymbol{\mathcal{I}}_R\|_F^2$ along factor matrix $\boldsymbol{A}$ should be computed with respect to $\text{vec}(\boldsymbol{A})$. However it is clear that $\frac{\partial f(\boldsymbol{A})}{\partial \text{vec}(\boldsymbol{A})} = \text{vec}\left(\frac{\partial f(\boldsymbol{A})}{\partial \boldsymbol{A}}\right)$ where f can be any function mapping to $\mathbb{R}$. This means that the derivative of the log-likelihood can be computed with respect to each factor matrix, which leads to the usual derivatives used in alternating least squares:

$$\begin{aligned} \frac{\partial \log(p(\boldsymbol{\mathcal{T}}|\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}))}{\partial \text{vec}(\boldsymbol{A})} &= \frac{1}{\sigma^2}\text{vec}\left(\frac{\partial \|\boldsymbol{\mathcal{T}} - (\boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{C})\boldsymbol{\mathcal{I}}_R\|_F^2}{\partial \boldsymbol{A}}\right) & \text{(B.4)} \\ &= \frac{1}{\sigma^2}\left(\boldsymbol{I}_K \boxtimes (\boldsymbol{B} \odot \boldsymbol{C})^{\mathsf{T}}\right)\text{vec}\left(\boldsymbol{T}_{(1)} - \boldsymbol{A}(\boldsymbol{B} \odot \boldsymbol{C})^{\mathsf{T}}\right) & \text{(B.5)} \end{aligned}$$

The right-hand term $\text{vec}\left(\boldsymbol{T}_{(1)} - \boldsymbol{A}(\boldsymbol{B} \odot \boldsymbol{C})^{\mathsf{T}}\right)$ is exactly the random variable $\text{vec}(\boldsymbol{E}_{(1)})$, the vectorized version of the matricized noise tensor $\boldsymbol{\mathcal{E}}$. To obtain the FIM, it is thus necessary to compute the expectation of the products of unfolded noise tensors. The FIM is a bloc matrix, and diagonal blocs can be obtained as follows. For the diagonal bloc corresponding to $\boldsymbol{A}$ $\boldsymbol{F}_{AA}$,

$$\begin{aligned} \boldsymbol{F}_{AA} &= \tfrac{1}{\sigma^4}\left(\boldsymbol{I}_K \boxtimes (\boldsymbol{B} \odot \boldsymbol{C})^{\mathsf{T}}\right)\mathbb{E}\left[\text{vec}(\boldsymbol{E}_{(1)})\text{vec}(\boldsymbol{E}_{(1)})^{\mathsf{T}}\right](\boldsymbol{I}_K \boxtimes (\boldsymbol{B} \odot \boldsymbol{C})) \\ &= \tfrac{1}{\sigma^2}\left(\boldsymbol{I}_K \boxtimes (\boldsymbol{B} \odot \boldsymbol{C})^{\mathsf{T}}\right)(\boldsymbol{I}_K \boxtimes (\boldsymbol{B} \odot \boldsymbol{C})) \\ &= \tfrac{1}{\sigma^2}\left(\boldsymbol{I}_K \boxtimes \left(\boldsymbol{B}^{\mathsf{T}}\boldsymbol{B} \boxdot \boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}\right)\right) \end{aligned} \tag{B.6}$$

However the blocs that are not on the diagonal of the Fisher matrix require the computation of $\mathbb{E}\left[\text{vec}(\boldsymbol{E}_{(i)})\text{vec}(\boldsymbol{E}_{(j)})^{\mathsf{T}}\right]$ where $i \neq j$. This calls for a permutation operator $\boldsymbol{P}_{ij}$ so that $\boldsymbol{P}_{ij}\text{vec}(\boldsymbol{T}_{(i)}) = \text{vec}(\boldsymbol{T}_{(j)})$. $\boldsymbol{P}_{ij}$ can be obtained by computation. Finally, the Fisher information matrix can be expressed as follows:

$$\boldsymbol{F} = \frac{1}{\sigma^2}\begin{bmatrix} \left(\boldsymbol{I}_K \boxtimes \left(\boldsymbol{B}^{\mathsf{T}}\boldsymbol{B} \boxdot \boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}\right)\right) & \left(\boldsymbol{I}_K \boxtimes (\boldsymbol{B}\odot\boldsymbol{C})^{\mathsf{T}}\right)\boldsymbol{P}_{12}\left(\boldsymbol{I}_L \boxtimes (\boldsymbol{A}\odot\boldsymbol{C})\right) & \left(\boldsymbol{I}_K \boxtimes (\boldsymbol{A}\odot\boldsymbol{B})^{\mathsf{T}}\right)\boldsymbol{P}_{13}\left(\boldsymbol{I}_M \boxtimes (\boldsymbol{A}\odot\boldsymbol{B})\right) \\ \left(\boldsymbol{I}_L \boxtimes (\boldsymbol{A}\odot\boldsymbol{C})^{\mathsf{T}}\right)\boldsymbol{P}_{12}^{\mathsf{T}}\left(\boldsymbol{I}_L \boxtimes (\boldsymbol{B}\odot\boldsymbol{C})\right) & \left(\boldsymbol{I}_L \boxtimes \left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} \boxdot \boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}\right)\right) & \left(\boldsymbol{I}_L \boxtimes (\boldsymbol{A}\odot\boldsymbol{C})^{\mathsf{T}}\right)\boldsymbol{P}_{23}\left(\boldsymbol{I}_M \boxtimes (\boldsymbol{A}\odot\boldsymbol{B})\right) \\ \left(\boldsymbol{I}_M \boxtimes (\boldsymbol{A}\odot\boldsymbol{B})^{\mathsf{T}}\right)\boldsymbol{P}_{13}^{\mathsf{T}}\left(\boldsymbol{I}_K \boxtimes (\boldsymbol{B}\odot\boldsymbol{C})\right) & \left(\boldsymbol{I}_M \boxtimes (\boldsymbol{A}\odot\boldsymbol{B})^{\mathsf{T}}\right)\boldsymbol{P}_{23}^{\mathsf{T}}\left(\boldsymbol{I}_L \boxtimes (\boldsymbol{A}\odot\boldsymbol{C})\right) & \left(\boldsymbol{I}_M \boxtimes \left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} \boxdot \boldsymbol{B}^{\mathsf{T}}\boldsymbol{B}\right)\right) \end{bmatrix} \tag{B.7}$$

Appendix C

Proof of existence of compressed non-negative factors

Proposition 5 *Let $\boldsymbol{A} \in \mathbb{R}^{m \times R}$, $m \geqslant R$ be a non-negative matrix, and $\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_R \end{bmatrix} \in \mathbb{R}^{m \times n}$ be the matrix of its first n left singular vectors, $n \leqslant R$, obtained from an SVD.*

1. *If $\boldsymbol{u}_1 \geq 0$, then the cone*
$$\mathcal{K} := \{\boldsymbol{v} \in \mathbb{R}^n \,|\, \boldsymbol{U}\boldsymbol{v} \geq 0\}$$
contains at least one nonzero vector.

2. *If $\boldsymbol{u}_1 > 0$, then the cone $\mathcal{K}$ is* solid *(has nonempty interior), i.e.,*
$$\operatorname{span} \mathcal{K} = \mathbb{R}^n.$$

The proof goes as follows.

1. Let $e_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^\top$. Since $\boldsymbol{u}_1 \geq 0$, we have that $\boldsymbol{U}e_1 \geq 0$.

2. The dual cone of the cone $\mathcal{K}$ is equal to
$$\mathcal{K}^* = \{\boldsymbol{U}^\top \boldsymbol{\alpha} \,|\, \boldsymbol{\alpha} \in \mathbb{R}^m, \boldsymbol{\alpha} \geq 0\}.$$
Since $\boldsymbol{u}_1 > 0$, we have that
$$\boldsymbol{v} \in \mathcal{K}^*, (\boldsymbol{v})_1 \leqslant 0 \Rightarrow \boldsymbol{\alpha} = 0 \Rightarrow \boldsymbol{v} = 0.$$
Therefore,
$$\mathcal{K}^* \cap -\mathcal{K}^* = \{0\},$$
i.e., the cone $\mathcal{K}^*$ is pointed[1].
As shown, for example in [13, p. 53], a dual cone of a pointed cone is a solid cone. □

Corollary 3 *1. For a random non-negative tensor $\boldsymbol{\mathcal{T}}$, the set $\left\{\boldsymbol{A}_c \in \mathbb{R}^{n \times R} | \boldsymbol{A} = \boldsymbol{U}\boldsymbol{A}_c\right\}$ is nonempty.*

2. Generically, the matrices $\boldsymbol{A}_c, \boldsymbol{B}_c, \boldsymbol{C}_c$ in $\left\{\boldsymbol{A}_c \in \mathbb{R}^{n \times R} | \boldsymbol{A} = \boldsymbol{U}\boldsymbol{A}_c\right\}$ have rank R.

The statements 1 and 2 follow from the corresponding statements of Proposition 5, because $\boldsymbol{U}$ (resp. $\boldsymbol{V}$ and $\boldsymbol{W}$) is the matrix of left singular vectors of the first (resp. second and third) unfolding matrix of $\boldsymbol{\mathcal{T}}$.

[1] Alternatively, $\mathcal{K}$ is a pointed cone if $\mathcal{K}\backslash 0$ lies in an open half space, *i.e.* there exists a hyperplane that intersects $\mathcal{K}$ only at 0.

Appendix D

Signal to Noise Ratios

In this appendix, we present the expressions for the SNR of two tensors $\boldsymbol{\mathcal{T}}_1$ and $\boldsymbol{\mathcal{T}}_2$ following the hybrid Gaussian models of Chapter 4. We present the SNR for the direct coupling model with unnormalized and normalized factors.

Direct coupling we recall briefly the models: the entries factors $\boldsymbol{A}_1$, $\boldsymbol{B}_1$, $\boldsymbol{A}_2$, $\boldsymbol{B}_2$ and $\boldsymbol{C}_2$ are drawn independently according to $\mathcal{N}(0,1)$, the noise tensors $\boldsymbol{\mathcal{E}}_1$ and $\boldsymbol{\mathcal{E}}_2$ are drawn from i.i.d. $\mathcal{N}(0,\sigma_1^2)$ and $\mathcal{N}(0,\sigma_2^2)$ respectively. The columns of factors are divided by their respective ℓ_2 norms in the normalized case. The elements of $\boldsymbol{C}_1$ are drawn from the elements of $\boldsymbol{C}_2$ using a zero mean Gaussian distribution of variance σ_c^2.

In the unnormalized case, the SNR for tensor $\boldsymbol{\mathcal{T}}_2$ writes as follows:

$$\mathrm{SNR}(\boldsymbol{\mathcal{T}}_2) = 10\log_{10}\left(\frac{R}{\sigma_2^2}\right) \tag{D.1}$$

while tensor $\boldsymbol{\mathcal{T}}_1$ has a more unusual SNR given by:

$$\mathrm{SNR}(\boldsymbol{\mathcal{T}}_1) = 10\log_{10}\left(\frac{R(1+\sigma_c^2)}{\sigma_1^2}\right) \tag{D.2}$$

In the normalized case we have

$$\mathrm{SNR}(\boldsymbol{\mathcal{T}}_1) = 10\log_{10}\left(\frac{R(1+\sigma_c^2)}{K_2L_2\sigma_1^2}\right) \tag{D.3}$$

and

$$\mathrm{SNR}(\boldsymbol{\mathcal{T}}_2) = 10\log_{10}\left(\frac{R}{K_2L_2\sigma_2^2}\right) \tag{D.4}$$

Note that for $r < R$ shared components, $\mathrm{SNR}(\boldsymbol{\mathcal{T}}_1)$ changes to $10\log_{10}\left[\frac{r\sigma_c^2+R}{KL\sigma_1^2}\right]$.